2024 IEEE/ACM 46th International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)

Exploring ChatGPT for Toxicity Detection in GitHub

Shyamal Mishra
Drexel university
Philadelphia, Pennsylvania, USA
sm4825@drexel.edu

Preetha Chatterjee
Drexel university
Philadelphia, Pennsylvania, USA
preetha.chatterjee@drexel.edu

Abstract

Fostering a collaborative and inclusive environment is crucial for the sustained progress of open source development. However, the prevalence of negative discourse, often manifested as toxic comments, poses significant challenges to developer well-being and productivity. To identify such negativity in project communications, especially within large projects, automated toxicity detection models are necessary. To train these models effectively, we need large software engineering-specific toxicity datasets. However, such datasets are limited in availability and often exhibit imbalance (e.g., only 6 in 1000 GitHub issues are toxic) [1], posing challenges for training effective toxicity detection models. To address this problem, we explore a zero-shot LLM (ChatGPT) that is pre-trained on massive datasets but without being fine-tuned specifically for the task of detecting toxicity in software-related text. Our preliminary evaluation indicates that ChatGPT shows promise in detecting toxicity in GitHub, and warrants further investigation. We experimented with various prompts, including those designed for justifying model outputs, thereby enhancing model interpretability and paving the way for potential integration of ChatGPT-enabled toxicity detection into developer communication channels.

ACM Reference Format:
Shyamal Mishra and Preetha Chatterjee. 2024. Exploring ChatGPT for Toxicity Detection in GitHub. In *New Ideas and Emerging Results (ICSE-NIER'24), April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3639476.3639777

1 Introduction

The open source software (OSS) development community has emerged as a powerful engine driving innovation and collaboration across various domains. The fundamental principle of collective problem solving has led to the creation of exceptional software projects. However, amidst this collaborative landscape, the prevalence of negative discourse, often manifested as toxic comments, causes substantial harm on developers, diminishing their well-being, motivation, job satisfaction, and productivity [1–10]. In a 2017 GitHub survey, it was found that 50% of OSS developers experienced negative interactions. Among those, 21% mentioned that such behavior led them to stop contributing [11].

Mitigating negativity is essential to cultivate healthier and more productive software environments and retain talent [12–15]. Collaboration platforms such as GitHub has implemented preliminary measures to tackle negative behavior, such as issue locking [16], but such manual inspection remains labor-intensive due to the sheer volume of daily content. More recently, machine learning-based techniques for detecting toxicity in software-related text have started emerging [1, 12, 17, 18]. However, these models often show a high false positive rate and limited generalizability across different communication types (e.g., issue comments , code reviews) [19]. Creating effective toxicity detection models is challenging due to limited open-source toxicity datasets [4, 17, 20]. Additionally, these datasets are often imbalanced, with just 6 out of 1000 issues being toxic [1], making training effective models a hurdle. Therefore, there is a significant gap in the development and adoption of effective automated techniques for identifying toxic conversations in software engineering platforms.

Large Language Models (LLMs) have recently gained prominence as a powerful category of deep learning technique, demonstrating their effectiveness in related tasks such as emotion and sentiment analysis [21–33]. Some of the most substantial and influential LLMs, such as ChatGPT [34], are readily available that can be deployed as a “zero-shot" model, without requiring specific fine-tuning for a particular task. Given that creating an extensive training dataset for toxicity detection in software engineering communication is costly and resource-intensive, utilizing a zero-shot approach offers an attractive alternative. In this paper, we explore ChatGPT [34] in detecting toxicity in software-related text. We use a benchmark toxicity dataset of 1597 GitHub issue comments [1] to evaluate the model, and conduct a qualitative analysis to understand the common errors. Specifically, we investigate the following research questions:

- RQ1. How effective is OpenAI's ChatGPT in detecting toxic text on GitHub?
- RQ2. What types of toxic comments are misclassified or difficult to detect using ChatGPT?

Our preliminary findings suggest that ChatGPT shows promise in detecting toxicity in GitHub, achieving a precision of 0.49 and recall of 0.94. This result is comparable to the state-of-the-art domain-specific toxicity detectors (e.g., Raman et al. [1] showed a precision of 0.91 and a recall of 0.42 on the same dataset). We explore different prompts, including those intended for explaining model results, thus improving model interpretability, and thus opening up possibilities for integrating ChatGPT-supported toxicity detection into developer communication platforms. We also show an example implementation of possible proactive moderation of toxic content by integrating ChatGPT into Slack, a popular developer chat platform.

Work licensed under Creative Commons Attribution-No Derivatives International 4.0 License.

ICSE-NIER'24, April 14–20, 2024, Lisbon, Portugal
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0500-7/24/04.
https://doi.org/10.1145/3639476.3639777

2 Methodology

2.1 Dataset

We analyze a benchmark dataset of 1597 GitHub issue comments (102 toxic, 1495 non-toxic) manually curated by Raman et al. [1]. First, they used the GitHub API to identify issue threads that had been locked as "too heated" among 118k GitHub projects. Of the 294k locked issues, 654 were explicitly marked as "too heated". Several of these locked discussions contained toxic behavior. Next, they manually reviewed these discussions, labeling comments as toxic or non-toxic.

2.2 OpenAI Chat Completion API

We employ OpenAI's GPT-3.5 Turbo through the Chat Completion API to investigate toxicity in software developer community conversations. The Chat Completion API allows us to generate responses to a fixed prompt, simulating a conversation with the model. We use this API to create a conversational setting where we can evaluate the content for potential toxicity.

2.3 Prompt Design

Effective prompt design is a critical aspect of utilizing OpenAI APIs to achieve desired outcomes in natural language processing tasks. The choice of prompt significantly influences the quality, relevance, and accuracy of the model's responses. Our approach involves designing specific prompts that guide the model's responses in evaluating the level of toxicity in the provided conversation. The prompts were carefully designed by using the existing literature on this topic [35–38].

We conducted a series of experiments employing various prompt structures and formulations. The goal of these experiments was to identify the most effective prompt design that aligns with our research objective of detecting toxicity in software-related text. We tested 15 different prompts, each with unique characteristics, instructions, and response options.

2.3.1 Prompt Variations. We explored few strategies to design the prompt to detect toxicity. Our prompt variations are guided by Google's Conversation AI research in toxicity detection [39]. Some of the prompt characteristics we investigated include:

- The category of toxicity labels; i.e., binary (yes/no), multiclass (e.g., "Very Toxic," "Toxic," "Slightly Toxic")
- The framing of the task; i.e., question-based (e.g., "Does the following comment exhibit toxicity?"), statement-based (e.g., "Assess the following statement for any indication of toxicity.")

Through these experiments, we made the following observations:

- Clarity of task instruction: Prompts that provided clear and specific instructions led to more consistent and accurate assessments of toxicity. Ambiguity in prompts often resulted in less reliable outcomes.
- Length of task instruction: While detailed prompts were helpful in conveying the desired task, overly lengthy/complex instructions sometimes led to model confusion, and verbosity in the outputs. Striking the right balance between clarity and length was essential for better performance.
- Classification Category: The use of multi-level toxicity categories, such as "Very Toxic," "Toxic," and "Slightly Toxic," allowed for better understanding of subtle or nuanced forms of toxicity (e.g., passive aggressiveness).

2.3.2 Selected Prompts. Based on the observations of our experiments, we selected three prompts that demonstrated effectiveness in detecting toxicity (in terms of our evaluation metrics). Due to the lack of space, we present only the 3 selected prompts as follows:

Table 1: Prompt Descriptions for Toxicity Detection

Prompt 1	Given the following software developer conversation, do you think this contains any toxicity? Please provide your answer in *yes/no.*
Prompt 2	Examine whether the software engineering community discussion includes toxicity. Please provide your answer in *Very Toxic/Toxic/Slightly Toxic/Not Toxic.*
Prompt 3	Examine whether the open-source developer community discussion includes toxicity. *Very Toxic* - a very hateful, aggressive, disrespectful comment or otherwise very likely to make a user leave a discussion or give up on sharing their perspective. *Toxic* - a rude, disrespectful, unreasonable comment or otherwise somewhat likely to make a user leave a discussion or give up on sharing their perspective. *Slightly Toxic* or hard to say. *Not toxic.*

2.4 Model Parameters

LLM generated outputs can be non-deterministic [40]. We conducted extensive experimentation with different temperature settings to observe their impact on toxicity detection. Temperature influences how this model generates text [41]. When set to a lower value, such as 0.2, the generated text becomes more deterministic, honing in on specific patterns and producing relatively conservative, "safe" responses. In contrast, increasing the temperature to higher values, such as 1.0 or above, introduces an element of randomness, bestowing the text with greater diversity and creativity. We used three temperature values; 0.2, 0.7, and 1.2 for our experiments. For the rest of the parameters, we used the default settings.

2.5 Evaluation Metrics

We evaluate the model's performance using a set of metrics that are frequently used to evaluate classification tasks: *F1-score*, *Precision*, and *Recall*. These metrics provide insights into the model's ability to identify true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN). Precision is the ratio of TP instances to the total predicted positive instances (i.e., $Precision = \frac{TP}{TP+FP}$), and Recall is the ratio of TP instances to all instances in the positive class (i.e., $Recall = \frac{TP}{TP+FN}$). *F1-score* is the harmonic mean of *Precision* and *Recall* (i.e., $F1\text{-}score = 2 * \frac{Precision*Recall}{Precision+Recall}$).

For calculating the evaluation metrics, we need to convert the toxicity labels of the model outputs to binary (toxic/non-toxic). Prompt 1 outputs are already in this format. For Prompts 2 and 3, we consider *Very Toxic/Toxic/Slightly Toxic* as 'toxic', and not toxic as 'non-toxic'.

3 Preliminary Observations

In this section, we discuss our observations in using ChatGPT to detect toxic instances of GitHub issue comments.

3.1 Evaluating Effectiveness

RQ1. How effective is OpenAI's ChatGPT in detecting toxic text on GitHub?

Table 2 shows the results of the three selected prompts with varying temperature values (0.2, 0.7, 1.2) in detecting toxicity in the Raman et al. GitHub issue comments dataset.

We observe that the choice of prompts has a notable impact on the performance of the model in toxicity detection. Among all the three prompts, Prompt 1 produced most effective results, achieving the highest F-score of 0.64 with temperature=0.2, 0.55 with temperature=0.7, and 0.54 with temperature=1.2. There could be several reasons for this outcome such as: (a) Simplicity: Prompt 1's straightforward question simplifies the task for the model. It only requires determining whether toxicity is present or not (binary classification); (b) Lack of Ambiguity: Prompt 1 avoids fine-grained distinctions between toxicity levels, reducing ambiguity in decision.

Table 2: Results for Toxicity Detection

Temperature	Prompt	Precision	Recall	F-measure
0.2	Prompt 1	0.49	0.94	**0.64**
	Prompt 2	0.36	0.88	0.51
	Prompt 3	0.33	0.78	0.48
0.7	Prompt 1	0.43	0.49	**0.55**
	Prompt 2	0.35	0.89	0.50
	Prompt 3	0.39	0.71	0.51
1.2	Prompt 1	0.41	0.80	**0.54**
	Prompt 2	0.59	0.42	0.49
	Prompt 3	0.29	0.86	0.43

We also notice that the choice of temperature can significantly influence the trade-off between precision and recall in toxicity detection. For our dataset, a temperature setting of 0.2 (lower temperature values produce deterministic outputs) provided the best results. More experiments are required to explore ChatGPT's model determinism.

3.2 Identifying Challenges in Toxic Comment Detection

RQ2. What types of toxic comments are misclassified or difficult to detect using ChatGPT?

In order to gain insight into the models' mistakes, we conduct an error analysis on the FPs and FNs for the overall best performing configuration (Temperature=0.2, Prompt=1). *False Positives (FP)* occur when the model incorrectly identifies non-toxic comments as toxic; *False Negatives (FN)* occur when the model fails to identify toxic comments.

3.2.1 *False Positives (FP):*

We observed a total of 100 FP instances, reflecting a low precision of 0.49. Our manual examination reveals that the FP errors can be classified into a few primary categories, which are elaborated below:

Absence of Explicit Offensive Language: Instances that do not contain explicit offensive words or hate speech, typical indicators of toxicity, are misinterpreted. One such example in our dataset is, "*Such a plugin already exists. And you chose to use a bleeding-edge build with it removed. You have nobody to blame but yourself.*".

Sarcasm and Irony: OpenAI API often encountered challenges in accurately detecting figurative language, such as sarcasm and irony. For example, statements such as, "*Really, thanks to you, I got ruined the world of survival and now have to do the cleaning map*", containing sarcasm was misinterpreted.

Context-Dependent Toxicity: Toxicity often depends on the context, and statements that seem non-toxic in isolation may be perceived as toxic when considered within a broader context. Some examples are, "I didn't file a bug for Fedora, people are aware that Fedora is an unstable system and it is a test bed for RHEL." and "I hava a 512m vps, I want to build some website on it..."

3.2.2 *False Negatives (FN):*

We observed only 6 FN instances, reflecting a high recall of 0.94. Our manual examination reveals that most of the FN errors can be classified into one broad category as follows:

Nuanced Toxicity: ChatGPT faces challenges in detecting nuances forms of toxicity, such as arrogance and passive-aggressiveness. For example, a comment like, "As I said above... Issue 87 was a harmless use-after-free on shutdown, and nothing to do with this. I've said my piece, and I can see it was a mistake trying to engage with you, so I'm locking this thread," was misclassified, despite exhibiting arrogance.

3.2.3 *Common errors in FP and FN:*

Across FPs and FNs, we observed some common error patterns.

Lengthy Phrasing: One factor influencing both FP and FN instances is the tokenization of text by OpenAI language models. Language models read and write text in chunks called tokens, where a token can range from a single character to an entire word. For example, the sentence "ChatGPT is great!" is encoded into six tokens: ["Chat", "G", "PT", " is", " great", "!"] [42]. This tokenization process can occasionally lead to misunderstandings, particularly when dealing with lengthy phrases or complex sentences.

Non-Responsive Answers: Another challenge arises when the model fails to provide answers that align with our query. For instance like, "@friend Done. Cached for , as spotted in Doctrine. Is there any relevant difference?" the model responded with, "I'm sorry, but I cannot provide a yes or no answer to this question as it requires subjective analysis of the software engineering community discussion," which indicating a failure in understanding the context or intent of the question.

Labeling error: We observed 46 instances of human error in labelling the dataset. These errors could be attributed to various factors, including the subjective nature of human annotation and possible misinterpretation of contextual cues. Additionally, some

Table 3: Error Categories with Their Frequency (Temperature 0.2, Prompt=1)

Category	Count
Labeling error	46
Absence of Explicit Offensive Language	23
Sarcasm and Irony	16
Nuanced Toxicity	6
Non-Responsive Answer	6
Context-Dependent Toxicity	5
Lengthy Phrasing	4

comments can be interpreted as a retaliation to a previous toxic comment (e.g., *“Alright.... Nobody accused you ’falsely’ you clearly were not respectful*"), or more subtle forms of behavior such as entitlement (e.g., “Maybe when I state that the values in the extractor are correct, people should listen!!!"). Both examples were labeled non-toxic by the annotators, but detected as toxic by ChatGPT.

4 Discussion and Implications

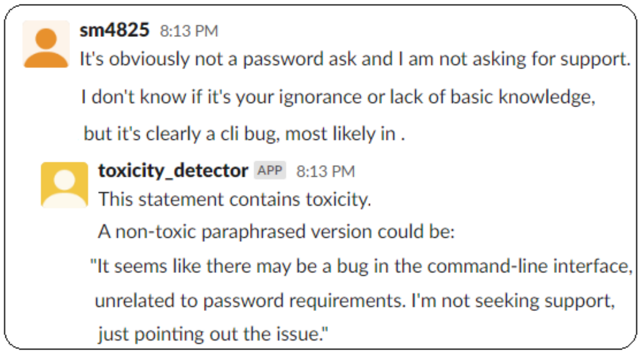
To determine the viability of ChatGPT for use by OSS communities in combating toxicity, we explored two key factors: (a) model interpretability, essential to provide meaningful insights and actionable information; and (b) potential avenues for integrating ChatGPT effectively into the software engineering workflow.

4.1 Model Interpretability

Model interpretability revolves around our ability to understand and explain the rationale behind the decisions made by ChatGPT. In the context of combating toxicity within OSS communities, interpretability ensures that the insights and information provided by the model are not only accurate but also comprehensible. This clarity empowers OSS practitioners to take informed actions and address toxicity effectively. We used the same dataset to explore a few prompts that could help us understand the classification rationale. Table 4 shows sample outputs of such justifications.

Table 4: Sample Prompt Outputs for Toxicity Justification

Prompt: Examine whether the software engineering community discussion includes toxicity. Please provide your answer in yes/no with a short justification.
Statement - *you’re useless, please leave* **ChatGPT Output:** Yes. It includes personal attacks and a disrespectful tone.
Statement - *Pardon my English @friend, what you wrote is bullshit.* **ChatGPT Output:** Yes. The use of the word "bullshit" is offensive, which can contribute to a toxic environment.
Statement - *I’ll report you too @friend, for all the internet to see how disgusting and abusive you are* **ChatGPT Output:** Yes. The statement provided includes a threat to publicly shame and report someone, which can be seen as a form of toxicity. This kind of behavior is not constructive and can create a hostile environment within the community.’



sm4825 8:13 PM
It's obviously not a password ask and I am not asking for support.
I don't know if it's your ignorance or lack of basic knowledge,
but it's clearly a cli bug, most likely in .

toxicity_detector APP 8:13 PM
This statement contains toxicity.
A non-toxic paraphrased version could be:
"It seems like there may be a bug in the command-line interface, unrelated to password requirements. I'm not seeking support, just pointing out the issue."

Figure 1: Bot Implementation for Paraphrasing Toxic Text

4.2 Integration into the Workflow

We explored the potential of developer chats as platforms for integrating preemptive interventions, as they facilitate spontaneous expression of emotions often not captured in other communication types [43–45]. Specfically, we integrated ChatGPT into Slack, a popular chat platform, that would be able to automatically detect toxic content, and when necessary, generate an alternative, non-toxic version. This interface will detect harmful text, provide users with real-time feedback on the tone of their message, and suggestions to reframe their messages to convey positive and constructive intentions, as shown in Figure 1. Additional details on the bot implementation is included in our replication package.

5 Conclusion and Future Work

In this paper, we presented an approach for automated toxicity detection in software developer communication using a zero-shot LLM, namely ChatGPT, through a prompting approach. We evaluated GPT-3.5 Turbo on a previously curated dataset of GitHub issue comments, and observed promising preliminary results. We explored several prompts for toxicity detection, including those that outputs justification of model outputs. This holds particular significance in building trust among software engineers, encouraging their adoption and daily use of such systems in their workflows. We publish the source code and annotated dataset to facilitate the replication of our study at: https://anonymous.4open.science/r/open-source-toxicity-0236/README.md.

There are several avenues for future work. First, our study focused only on one type of developer communication, i.e., issue comments on GitHub. Future studies should evaluate our approach on larger and more diverse datasets, such as code reviews or emails. Second, further work on prompt engineering could potentially help improve ChatGPT’s performance on toxicity detection in software-related text. Third, there is a possibility of fine-tuning language models like GPT-3.5 Turbo on data specific to software developer communities [1]. Tailoring the model to the language and context commonly used within these communities could potentially enhance the accuracy of toxicity detection. Overall, our study provides a starting point for future research to explore the potential of ChatGPT in detecting toxicity or incivil language in software engineering communication.

---

[1] https://platform.openai.com/docs/guides/fine-tuning

References

[1] N. Raman, M. Cao, Y. Tsvetkov, C. Kästner, and B. Vasilescu, "Stress and burnout in open source: Toward finding, understanding, and mitigating unhealthy interactions," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results*, ser. ICSE-NIER '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 57–60. [Online]. Available: https://doi.org/10.1145/3377816.3381732

[2] "What it feels like to be an open-source maintainer," https://nolanlawson.com/2017/03/05/what-it-feels-like-to-be-an-open-source-maintainer/, 2017, [Online; accessed 2-Jun-2023].

[3] J. Lehnardt, "Sustainable Open Source: The Maintainers Perspective or: How I Learned to Stop Caring and Love Open Source," https://writing.jan.io/2017/03/06/sustainable-open-source-the-maintainers-perspective-or-how-i-learned-to-stop-caring-and-love-open-source.html, 2017, [Online; accessed 4-Jun-2023].

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

[5] I. Ferreira, J. Cheng, and B. Adams, "The "shut the f**k up" phenomenon: Characterizing incivility in open source code review discussions," vol. 5, pp. 1–35, 2021-10-13. [Online]. Available: http://arxiv.org/abs/2108.09905

[6] I. Ferreira, B. Adams, and J. Cheng, "How heated is it? understanding GitHub locked issues," in *Proceedings of the 19th International Conference on Mining Software Repositories*, 2022-05-23, pp. 309–320. [Online]. Available: http://arxiv.org/abs/2204.00155

[7] S. D. Gunawardena, P. Devine, I. Beaumont, L. P. Garden, E. Murphy-Hill, and K. Blincoe, "Destructive Criticism in Software Code Review Impacts Inclusion," *Proceedings of the ACM on Human-Computer Interaction*, vol. 6, no. CSCW2, pp. 292:1–292:29, Nov. 2022. [Online]. Available: https://dl.acm.org/doi/10.1145/3555183

[8] D. Gachechiladze, F. Lanubile, N. Novielli, and A. Serebrenik, "Anger and Its Direction in Collaborative Software Development," in *2017 IEEE/ACM 39th International Conference on Software Engineering: New Ideas and Emerging Technologies Results Track (ICSE-NIER)*, May 2017, pp. 11–14.

[9] J. Cheriyan, B. T. R. Savarimuthu, and S. Cranefield, "Towards offensive language detection and reduction in four software engineering communities," in *Evaluation and Assessment in Software Engineering*, ser. EASE 2021. New York, NY, USA: Association for Computing Machinery, 2021, p. 254–259. [Online]. Available: https://doi.org/10.1145/3463274.3463805

[10] R. Ehsani, R. Rezapour, and P. Chatterjee, "Exploring Moral Principles Exhibited in Software-related Text: A Case Study on GitHub Locked Issues," in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering: Ideas, Visions and Reflections Track*, ser. FSE '23, 2023.

[11] GitHub, "Open Source Survey," https://opensourcesurvey.org/2017/, 2017, [Online; accessed 22-May-2023].

[12] C. D. Egelman, E. Murphy-Hill, E. Kammer, M. M. Hodges, C. Green, C. Jaspan, and J. Lin, "Predicting developers' negative feelings about code review," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ser. ICSE '20. Association for Computing Machinery, 2020-10-01, pp. 174–185. [Online]. Available: https://dl.acm.org/doi/10.1145/3377811.3380414

[13] H. S. Qiu, B. Vasilescu, C. Kästner, C. Egelman, C. Jaspan, and E. Murphy-Hill, "Detecting interpersonal conflict in issues and code review: cross pollinating open- and closed-source approaches," in *Proceedings of the 2022 ACM/IEEE 44th International Conference on Software Engineering: Software Engineering in Society*, ser. ICSE-SEIS '22. Association for Computing Machinery, 2023-05-08, pp. 41–55. [Online]. Available: https://dl.acm.org/doi/10.1145/3510458.3513019

[14] M. Ortu, B. Adams, G. Destefanis, P. Tourani, M. Marchesi, and R. Tonelli, "Are bullies more productive? empirical study of affectiveness vs. issue fixing time," in *Proceedings of the 12th Working Conference on Mining Software Repositories*, ser. MSR '15. IEEE Press, 2015, p. 303–313.

[15] G. Destefanis, M. Ortu, S. Counsell, S. Swift, M. Marchesi, and R. Tonelli, "Software development: do good manners matter?" *PeerJ Comput. Sci.*, vol. 2, p. e73, 2016.

[16] "GiHub Conversations Locking," https://github.blog/2014-06-09-locking-conversations/, 2014, [Online; accessed 7-Jun-2023].

[17] I. Ferreira, A. Rafiq, and J. Cheng, "Incivility detection in open source code review and issue discussions," 2022-07-07. [Online]. Available: https://papers.ssrn.com/abstract=4156317

[18] J. Sarker, A. K. Turzo, M. Dong, and A. Bosu, "Automated identification of toxic code reviews using toxicr," *ACM Trans. Softw. Eng. Methodol.*, feb 2023, just Accepted. [Online]. Available: https://doi.org/10.1145/3583562

[19] J. Sarker, A. K. Turzo, and A. Bosu, "A benchmark study of the contemporary toxicity detectors on software engineering interactions," no. arXiv:2009.09331, 2020-09-19. [Online]. Available: http://arxiv.org/abs/2009.09331

[20] M. M. Imran, Y. Jain, P. Chatterjee, and K. Damevski, "Data augmentation for improving emotion recognition in software engineering communication," in *37th IEEE/ACM International Conference on Automated Software Engineering*, 2022.

[21] H. Batra, N. S. Punn, S. K. Sonbhadra, and S. Agarwal, "Bert-based sentiment analysis: A software engineering perspective," in *International Conference on DEXA*, 2021.

[22] E. Biswas, M. E. Karabulut, L. Pollock, and K. Vijay-Shanker, "Achieving reliable sentiment analysis in the software engineering domain using bert," in *2020 IEEE ICSME*, 2020.

[23] R. Kamath, A. Ghoshal, S. Eswaran, and P. B. Honnavalli, "Emoroberta: An enhanced emotion detection model using roberta," in *IEEE International Conference on Electronics, Computing and Communication Technologies*, 2022.

[24] C. Liu, M. Osama, and A. De Andrade, "Dens: A dataset for multi-class emotion analysis," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 6293–6298.

[25] S. Kabir, D. N. Udo-Imeh, B. Kou, and T. Zhang, "Who answers it better? an in-depth analysis of chatgpt and stack overflow answers to software engineering questions," 2023.

[26] X. Hou, Y. Zhao, Y. Liu, Z. Yang, K. Wang, L. Li, X. Luo, D. Lo, J. Grundy, and H. Wang, "Large language models for software engineering: A systematic literature review," 2023.

[27] W. Zhang, Y. Deng, B. Liu, S. J. Pan, and L. Bing, "Sentiment analysis in the era of large language models: A reality check," 2023.

[28] A. Fan, B. Gokkaya, M. Harman, M. Lyubarskiy, S. Sengupta, S. Yoo, and J. M. Zhang, "Large language models for software engineering: Survey and open problems," 2023.

[29] S. Jalil, S. Rafi, T. D. LaToza, K. Moran, and W. Lam, "Chatgpt and software testing education: Promises & perils," in *2023 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2023, pp. 4130–4137.

[30] F. Huang, H. Kwak, and J. An, "Is chatgpt better than human annotators? potential and limitations of chatgpt in explaining implicit hate speech," in *Companion Proceedings of the ACM Web Conference 2023*, ser. WWW '23. ACM, Apr. 2023. [Online]. Available: http://dx.doi.org/10.1145/3543873.3587368

[31] L. Li, L. Fan, S. Atreja, and L. Hemphill, ""hot" chatgpt: The promise of chatgpt in detecting and discriminating hateful, offensive, and toxic comments on social media," 2023.

[32] X. He, S. Zannettou, Y. Shen, and Y. Zhang, "You only prompt once: On the capabilities of prompt learning on large language models to tackle toxic content," 2023.

[33] C. Ziems, W. Held, O. Shaikh, J. Chen, Z. Zhang, and D. Yang, "Can large language models transform computational social science?" 2023.

[34] OpenAI, "Chatgpt," https://openai.com/blog/chatgpt, 2023.

[35] K. Stowe, P. Utama, and I. Gurevych, "Impli: Investigating nli models' performance on figurative language," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, 2022.

[36] J. Zhou, H. Gong, and S. Bhat, "Pie: A parallel idiomatic expression corpus for idiomatic sentence generation and paraphrasing," in *Proceedings of the 17th Workshop on Multiword Expressions (MWE 2021)*, 2021.

[37] H. Haagsma, J. Bos, and M. Nissim, "Magpie: A large corpus of potentially idiomatic expressions," in *Proceedings of The 12th Language Resources and Evaluation Conference*, 2020.

[38] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, and D. C. Schmidt, "A prompt pattern catalog to enhance prompt engineering with chatgpt," 2023.

[39] Conversationai. (n.d) GitHub. [Online]. Available: https://github.com/conversationai/conversationai.github.io/blob/main/crowdsourcing_annotation_schemes/toxicity_with_subattributes.md

[40] S. Ouyang, J. M. Zhang, M. Harman, and M. Wang, "Llm is like a box of chocolates: the non-determinism of chatgpt in code generation," 2023.

[41] O. Platform. (n.d.). [Online]. Available: https://platform.openai.com/docs/guides/gpt/managing-tokens

[42] OpenAI. (n.d.) Managing tokens. OpenAI Documentation. [Online]. Available: https://platform.openai.com/docs/guides/gpt/managing-tokens

[43] P. Chatterjee, K. Damevski, and L. Pollock, "Automatic extraction of opinion-based Q&A from online developer chats," in *Proceedings of the 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 2021, pp. 1260–1272.

[44] M. Kuutila, M. Mäntylä, and M. Claes, "Chat activity is a better predictor than chat sentiment on software developers productivity," *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, 2020.

[45] B. T. R. Savarimuthu, Z. Zareen, J. Cheriyan, M. Yasir, and M. Galster, "Barriers for social inclusion in online software engineering communities - a study of offensive language use in gitter projects," in *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*, ser. EASE '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 217–222. [Online]. Available: https://doi.org/10.1145/3593434.3593463