PRemo: A Dataset of Emotions Found on Pull Request Discussions

Daniel Coutinho
Pontifical Catholic University of Rio de Janeiro (PUC-Rio)
Rio de Janeiro, Brazil
dcoutinho@inf.puc-rio.br

Juliana Alves Pereira
Pontifical Catholic University of Rio de Janeiro (PUC-Rio)
Rio de Janeiro, Brazil
juliana@inf.puc-rio.br

Breno Braga Neves
Pontifical Catholic University of Rio de Janeiro (PUC-Rio)
Rio de Janeiro, Brazil
brenonevs@aise.inf.puc-rio.br

João Correia
Pontifical Catholic University of Rio de Janeiro (PUC-Rio)
Rio de Janeiro, Brazil
jcorreia@inf.puc-rio.br

Caio Barbosa
Pontifical Catholic University of Rio de Janeiro (PUC-Rio)
Rio de Janeiro, Brazil
csilva@inf.puc-rio.br

Wesley K.G. Assunção
North Carolina State University (NCSU)
Raleigh, USA
wguezas@ncsu.edu

Igor Steinmacher
Northern Arizona University (NAU)
Flagstaff, USA
Igor.Steinmacher@nau.edu

Marco Gerosa
Northern Arizona University (NAU)
Flagstaff, USA
Marco.Gerosa@nau.edu

Augusto Baffa
Pontifical Catholic University of Rio de Janeiro (PUC-Rio)
Rio de Janeiro, Brazil
abaffa@inf.puc-rio.br

Alessandro Garcia
Pontifical Catholic University of Rio de Janeiro (PUC-Rio)
Rio de Janeiro, Brazil
afgarcia@inf.puc-rio.br

ABSTRACT

In software engineering, effective communication is key to ensuring software quality. On GitHub projects, one of the primary channels for such communication is pull request discussions. These discussions often contain high-level design decisions. Understanding communication dynamics within development teams is crucial in software engineering, and sentiments and emotions play a significant role in this context. While there are datasets available for sentiment analysis in this field, those focusing on specific emotions are rare, and some contexts, such as pull request discussions, remain underrepresented. To address these gaps, we propose a novel methodology for capturing sentiments and emotions in context-specific data, resulting in the creation of PRemo. PRemo includes ≈1.8K manually labeled pull-request messages from 36 active open-source industry-relevant projects. It provides data on individual emotions (and their intensity), the surrounding context, and evaluator confidence. Built using a robust triple validation and two-pass labeling process, the dataset leverages an established psychological emotion model. Already applied in prior research, PRemo is a valuable resource for advancing emotion analysis in software engineering.

KEYWORDS

repository mining, human aspects, sentiment analysis

1 Introduction

Effective communication in software engineering (SE) is key to creating high-quality software systems [6, 7, 14, 34, 36, 40, 43]. As such, one key area of interest among researchers has been understanding the human aspects of this communication process. The analysis of these aspects can be used in various SE contexts, from interpreting feedback on code reviews [24, 28, 35, 38, 39] to assessing the mood of the discussions in a development team [18, 19].

One key facet of this research is aimed at analyzing and understanding the emotions and sentiments involved in this communication process. While sentiment analysis typically focuses on determining the general polarity of a message (i.e., whether it is positive, negative, or neutral), emotion analysis aims to detect specific affective states, such as joy, anger, or sadness. By applying these types of analyses, stakeholders can pinpoint areas of concern, while finding ways to enhance teamwork and ensuring that projects evolve smoothly [6, 15, 37].

In essence, leveraging sentiment and emotion analysis in SE opens the door to more nuanced interactions and a deeper understanding of project dynamics, which previous research has pointed out as a key factor affecting software quality [18]. For example, one study showed that negative sentiment in GitHub commit messages is significantly associated with the introduction of software bugs [20], while another found that both positive and certain negative emotions in developer chats were linked to higher productivity in terms of commits and lines of code [25]. Additionally, developers' emotions during code reviews have been tied to productivity fluctuations, with affective states influencing their task performance [5].

While a variety of datasets for sentiment analysis within SE have emerged [4, 8, 21, 22, 30, 32], datasets containing data about individual emotions are considerably rarer [16, 27]. Nevertheless, two other limitations can hinder their usage. First, these datasets are often constructed using different methodologies, which leads to varying levels of quality and inconsistency in the type of data collected. Second, datasets that focus on code review processes tend to emphasize comments anchored to specific lines of code, such as inline review comments. These comments, while essential for improving code quality, are typically highly technical and narrow in scope—focusing on specific issues (e.g, syntax errors, naming conventions, or bugs), which limits their suitability for emotion or sentiment analysis.

While on older platforms these comments were the main part of the code review, on GitHub, these code-level review comments coexist within broader discussions that occur on pull requests (PRs). These PR discussions are not limited to isolated code annotations but also include general conversations about the rationale behind changes, design decisions, or project direction. Although they can

also contain technical exchanges, these high-level discussions tend to be more diverse in content and tone, often reflecting disagreement, appreciation, frustration, or support. We refer to this broader conversational space as PR discussions. Given their centrality in collaborative software development and their more expressive and discursive nature, PR discussions offer a richer source of information for understanding interpersonal dynamics, sentiment, and emotion within development teams.

As such, the goal of this work is to design and create a novel dataset, which we call PRemo[1]. PRemo is intended to support automatic techniques in effectively capturing the emotions expressed by developers during PR discussions. To create PRemo, we designed a labeling process in which 19 evaluators evaluated 2,200 messages from 36 different GitHub projects. The entire labeling process used emotion and sentiment classes taken from psychology research [33].

To distinguish PRemo from existing datasets, we included several novel ideas in its design and creation process, aimed at enabling novel studies and reducing subjectivity in the labeling process. First, we employed a triple validation procedure, where each message was labeled by three evaluators, who also reported their confidence in their evaluations. To ensure diverse perspectives, each message was assessed by evaluators from two different areas: one neuroscientist[2] and two software engineers. Second, all experts performed a two-pass labeling process for each message. In the first pass, they evaluated the message in isolation, while on the second pass, they had access to the full context of the message (see Section 3). This two-pass methodology allowed us to analyze whether having access to more information influenced the experts' evaluations, providing valuable insights into the role of context in emotion perception. Finally, PRemo stands out as one of the few datasets that contains emotion-specific labels (e.g., fear or sadness), as opposed to merely indicating sentiment polarity (e.g., positive or negative). Additionally, the dataset includes data on the intensity of sentiment polarity, capturing how strongly negative or positive each expert considered a message. These features make PRemo a rich resource for advancing emotion analysis in SE.

In summary, the contributions of this paper are:

- A methodology for building a dataset on sentiment and emotion data, considering novel aspects such as (i) the importance of having the full context of a message, and (ii) the key goal of reducing subjectivity, by employing mechanisms such as triple validation and the usage of an emotion model well established in psychology [33].
- A dataset of ≈1.8K messages from 36 GitHub projects labeled by 19 evaluators from both software engineering and neuroscience backgrounds, providing a rich and multi-perspective annotation process. PRemo dataset is available at [13].
- The availability of the artifacts regarding the construction of the dataset, including raw annotation data (i.e., individual answers by the evaluators), scripts, and tools to support replication and reuse by the research community[13].
- A foundation basis for future research and tool development aimed at improving communication quality, fostering emotional awareness, and promoting the well-being of developers in collaborative software development environments.

2 Background and Related Work

The recent literature shows several studies focused on introducing tools for sentiment analysis in SE contexts [4, 8, 21, 22]. These studies rely on datasets containing messages from the SE contexts and their respective sentiment classifications [30, 32]. However, we are only aware of one sentiment dataset [9] which considers theoretical emotion models for classifications, and only two datasets that provide emotion data, instead of sentiment [10, 11]. While some of them may use GitHub data, none of them are in the context of PR discussions.

Islam et al. [21] proposed a domain dictionary based on JIRA issue comments. The authors do not mention using an existing conceptual emotion framework to build their dictionary. Although they did not employ Shaver's [33] emotion model, the messages were also classified into love, joy, surprise, anger, sadness, and fear. Calefato et al. [8] use three datasets to investigate emotions in Q&A platforms such as StackOverflow, Jira, and Github. The first comprises 4,423 StackOverflow posts manually annotated by twelve experts. The second includes 5,869 Jira sentences annotated by three experts. The third encompasses 7,000 sentences from GitHub annotated by three experts [30]. Although they employed Shaver's model [33], it is unclear whether their data includes PR discussions.

Ahmed et al. [4] presented a dataset of 2,000 review messages evaluated by three experts. The principal limitation of this work was that experts only annotated the polarity of messages (i.e., positive or negative). In addition, Islam et al., [22] presented a dataset containing 1,795 JIRA issue comments evaluated by three graduate students. The evaluators employed a bi-dimensional emotion model. In the first dimension, the evaluators classified the message as positive, negative, or neutral. Then, they classified as high or low arousal.

In this work, we aim to present a new dataset that overcomes the limitations observed in previous datasets used for sentiment analysis in SE. Our dataset incorporates a theoretical emotion model [33], which was already used in SE by [8]. This model builds upon the foundation of basic emotion theories, such as Ekman's [17], by organizing discrete emotions into a structured hierarchy. We targeted an important context that few datasets currently cover (i.e., PR discussions). In addition, we added novel procedures to the construction of our dataset, such as accounting for the context surrounding the messages, and using a triple validation (see Section 3).

3 Methodology

In this section, we detail the five steps of the method we used to collect and organize the data in PRemo. These steps are also shown in Figure 1.

3.1 Project Selection

To start building the dataset, we first selected the software projects from which we gathered the PR messages, following the inclusion criteria (IC):

[1]The name PRemo stands for Pull Request Emotions. The acronym emphasizes both the technical context (PRs) and the emotional dimension (emo), aligning with the dataset's goal of enabling sentiment and emotion analysis in SE communication.
[2]The neuroscientist selected had both familiarity with the emotion model used in this study and programming experience.

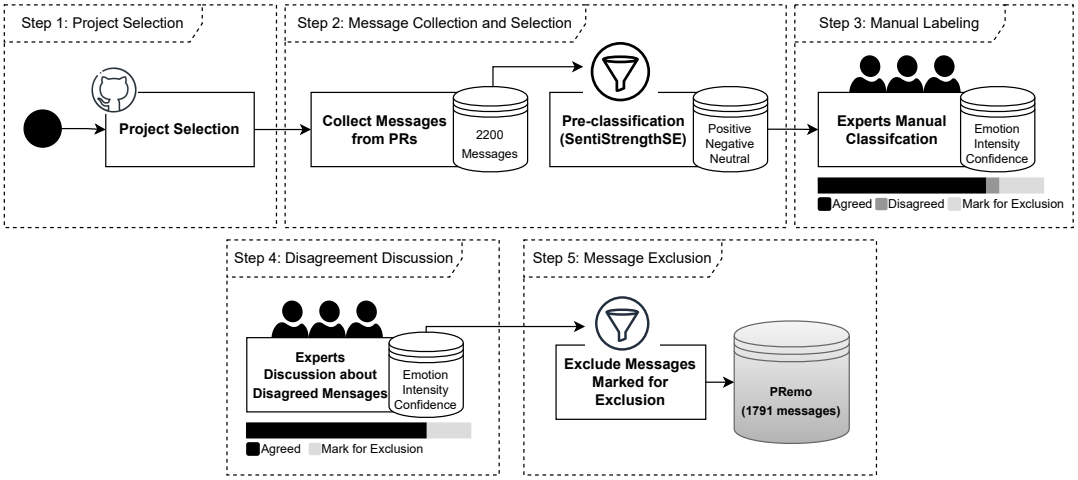

Figure 1: Overview of the Methodology

IC1 *The project has to be hosted on GitHub.* We decided to use GitHub since it is the largest and most popular code hosting platform, offering a vast repository of open-source software projects. Its widespread adoption by developers worldwide ensures a comprehensive and diverse dataset, making it an ideal environment for analyzing open-source software developers' interaction.

IC2 *The project must use a pull-based workflow.* One of the differences of PRemo is providing the analysis on the PR discussion level. Therefore, we only considered those projects that use the PR feature to receive, review, and merge contributions. To filter that, we performed a manual analysis of the candidate projects.

IC3 *The project should have at least 1,000 PRs.* This criterion aimed to filter out less active, and toy/personal projects. This enables us to concentrate on those projects with a significant development history.

IC4 *The project should have commits in the 3 months previous to the collection.* We applied this filter to ensure the analysis of projects that demonstrate ongoing development and engagement. Along with criterion (IC3), this helped us remove toy projects. Furthermore, we opted not to include dead projects, as we wanted our dataset to represent up-to-date development practices.

IC5 *The project should be at least five years old.* This was done to ensure the focus on mature projects, which are more likely to have gone through different development phases, and thus provide a more comprehensive view of long-term project maintenance, and community dynamics.

Although many projects met our criteria, we deliberately chose a limited set of 36 projects listed in Table 1. We chose to keep a manageable number of projects to ensure a well-balanced distribution of messages across projects, given the manual work required for the labeling of each PR. We decided to chose different projects based on the assumption that development practices can vary wildly between projects [6]. As such, we wanted to ensure that the projects in PRemo were diverse, to increase the representativeness of our dataset. Thus, beside the aforementioned criteria, we chose projects of different domains, communities, and sizes. Finally, we also chose projects that use, as their main programming languages, ones that are cited as being different from one another in their common application domains. [1–3]: Java, JavaScript, TypeScript, and Python. See Table 1 for more details about the projects.

3.2 Message Collection and Selection

Upon project selection, we used the GitHub API to gather all pull-request-level discussions from closed PRs across the entire history of each chosen project. Throughout this data collection activity, we filtered messages originating from automated bots, both by using the information about bots provided by the GitHub API itself and also with a simple heuristic filtering for names in which the word bot is prominent. This process led us to gather around one million messages.

Given the large number of messages in our dataset, the subsequent activity involved choosing messages that would undergo manual labeling by evaluators. Since previous research observed that technical messages are often neutral [23], we decided to utilize SentiStrengthSE [21] in a pre-classification step. SentiStrengthSE has been heavily used by previous works [31]. This process resulted in three pre-classified groups of messages: *positive*, *negative*, and *neutral*. The analysis was conducted in two rounds, and we sampled messages from all 36 projects at the end of the process. In the *first round*, 1,000 messages from 11 Java projects were randomly selected. As the messages were pre-classified, we chose to select a specific distribution of messages, based on the aforementioned assumption that the data would be biased towards neutral messages. This resulted in a group of 350 random positive messages, 350 random negative messages, and 300 random neutral messages. These groups are based in the pre-classification only and do not reflect the results of the manual labeling. The random selection utilized the entire pool of messages for these 11 projects, instead of a per-project approach.

Table 1: Projects Utilized in the Creation of the Dataset

Main Programming Language	Project	Domain	Created In	Age	LOC	# Pull Requests	# Contributors
Java	spring-projects/spring-boot	Development Framework	2012	12 years	~420k	6169	1074
	spring-projects/spring-security	Security Framework	2012	12 years	~445k	2846	694
	google/guice	Dependency Injection Framework	2014	10 years	~106k	625	74
	google/ExoPlayer	Library	2014	10 years	~479k	1191	239
	google/guava	Library	2014	10 years	~778k	2185	302
	google/gson	Library	2015	9 years	~53k	996	147
	google/dagger	Dependency Injection Framework	2013	11 years	~167k	2287	-
	netflix/eureka	Service Registry	2012	12 years	~84k	864	108
	netflix/hystrix	Fault Tolerance Library	2012	12 years	~78k	812	113
	netflix/conductor	Microservice	2016	8 years	~90k	1702	248
	netflix/zuul	API Gateway	2013	11 years	~73k	1195	57
	JabRef/jabref	Graphical Library	2014	10 years	~235k	6901	630
	mockito/mockito	Test Framework	2012	12 years	~97k	1669	288
JavaScript (or Typescript)	vuejs/core	JS Framework	2018	6 years	~125k	4271	455
	twbs/bootstrap	Web Framework	2011	13 years	~44k	15110	1390
	expressjs/express	Node Framework	2009	15 years	~23k	1273	307
	facebook/react	Web Framework	2013	11 years	~494k	14735	1656
	sveltejs/svelte	Web Application	2016	8 years	~84k	4594	670
	ant-design/ant-design	React Library	2015	9 years	~193k	16764	2091
	angular/angular	Web Framework	2014	10 years	~790k	26712	1882
	d3/d3	Web Library	2010	14 years	~20k	1170	132
	microsoft/TypeScript	Programming Language	2014	10 years	~3.4M	17314	771
	mrdoob/three.js	JS Library	2010	14 years	~426k	15603	1866
	jestjs/jest	Test Framework	2013	11 years	~120k	7165	1532
	puppeteer/puppeteer	Node API	2017	7 years	~76k	5428	485
Python	tiangolo/fastapi	Python Framework	2018	6 years	~109k	3161	633
	matplotlib/matplotlib	Python Library	2011	13 years	~249k	17823	1415
	tinygrad/tinygrad	Python Framework	2020	4 years	~93k	3354	296
	plotly/plotly.py	Python Library	2013	11 years	~902k	1617	238
	pandas-dev/pandas	Python Library	2010	14 years	~612k	31839	3168
	pydantic/pydantic	Python Library	2017	7 years	~109k	3467	507
	psf/requests	HTTP Library	2011	13 years	~11k	2490	642
	tensorflow/tensorflow	ML Framework	2015	9 years	~1.2M	25164	3530
	astropy/astropy	Library	2011	13 years	~382k	10300	485
	pallets/flask	Python Framework	2010	14 years	~17k	2524	715
	ansible/ansible	Framework	2012	12 years	~245k	50519	5000+

When manual labeling (see Section 3.3) the messages selected on the first round, we received feedback from the evaluators that the messages were too similar, and that might be because some projects might be under- or overrepresented. Then, we decided that our message randomization process for the second round would follow a per-project approach. Therefore, for the *second round*, we selected 1,200 messages from the 25 remaining projects in our sample. In this new approach, for each project, we selected 48 messages, being 20 random positive messages, 20 random negative messages, and eight random neutral messages. These exact numbers were selected as we had a target of 2,200 messages, in order to not overwhelm the evaluators, given the manual nature of the work, which could lower the quality of the results. While the distribution of messages was already skewed towards positive and negative messages in the first round, we observed that our dataset had a high frequency of neutral messages, even with the pre-classification process. Thus, for the second round, we purposefully unbalanced the classes and selected fewer neutral messages.

3.3 Manual Labeling

First, we manually labeled the sentiments of the 1,000 messages selected in the first round, as detailed above. This activity was performed by 19 evaluators. 16 were software engineering students—spanning from undergraduates to PhD candidates with diverse levels of industry experience. The other three evaluators were neuroscientists, chosen because they had familiarity with Shaver's emotion model [33], which we employed in this step. All of the neuroscientists were also familiar with programming.

While there are a myriad of theoretical emotion models that could be employed in sentiment analysis, we utilized Shaver's

model [33], since it has been previously utilized for sentiment analysis in the context of SE [26, 29]. This model is based on a hierarchy of emotions, divided into two polarities: positive and negative. Each emotion is also considered to have an intensity level. Shaver's model utilized six basic emotions: (i) *love* and (ii) *joy* that can be seen as positive emotions; (iii) *anger*, (iv) *sadness*, and (v) *fear* that can be seen as negative emotions; and (vi) *surprise*, in which polarity is context-dependent. While the model does not describe neutrality of emotion, previous works [8, 29] have described *neutral* messages in the context of SE, either as the presence of two opposite (in terms of polarity) but equally intense emotions, or the absence of any emotion. We utilized both types of neutral messages in this work.

To support evaluators when performing the manual labeling process, we utilized a tool developed by the authors specifically for this study[3]. This tool provides a guide explaining how to apply the theoretical emotion model to label each message, with several examples of pre-classified messages, to reduce subjectivity. All evaluators in the labeling process were asked to carefully read the guide, to facilitate the onboarding process and level the knowledge of all participants. The tool also enforced a triple validation model, where each message was labeled by two SE evaluators and one neuroscience student.

For each message, the evaluators had to first specify whether they identified any emotions in the message. If the answer was that at least one emotion exists in the message, they then specified which emotions from the model they identified, and the intensity of those emotions. For intensity, we utilized a 4-point scale representing each polarity (*i.e.*, whether each emotion polarity was slightly, moderately, very and extremely present). Finally, evaluators also had to report a confidence level, using a 5-point scale. We purposefully made each evaluator do this labeling process twice for each message. In the *first pass*, they only had access to the individual message, while on the *second pass*, we gave them access to the entire thread where the message was located. This was done to see if other contextual information available in the conversation thread (e.g., other messages and the identity of participants) would affect the labeling results.

3.4 Disagreement Discussion

Since each message was evaluated by three evaluators, the final label was assigned by the majority vote. We identified a few cases when this was not possible (i.e., there were positive, negative, and neutral labels indicated by each evaluator). This represented ≈3% (54) out of the 2,200 messages. For these cases, the evaluators were asked to discuss the details of the message via a chat thread on Discord[4]. After this discussion, they could change their original responses. For most messages (51 out of 54), the evaluators reached a consensus, and these labels were then included in the dataset[5]. The messages in which it was not possible to reach consensus were then removed from the dataset.

[3]Details regarding this tool can be found on our website [13].
[4]https://discord.com
[5]These messages are marked on the dataset, as their sentiment polarity is labeled as undefined. For these messages, the polarity originating from the discussions is contained in a special optional field.

3.5 Message Exclusion

Despite the actions taken on the step described in Section 3.2 to filter invalid messages (*i.e.*, messages automatically created by bots), the evaluators still identified problems with some messages (e.g., bots, duplicated messages, messages with no textual content, etc.), which was expected given that while the messages did pass through a filtering process, we did not filter them manually and utilized simple heuristics. Thus, as a final step, we employed the following criteria to manually filter the final set of messages:

(i) The message had to contain textual content (*i.e.*, we excluded messages containing only mentions or links). Emojis were considered textual content;
(ii) The message had to be written by a human (*i.e.*, we excluded messages created by bots);
(iii) The message had to be written in English; and,
(iv) Duplicated messages were not allowed.

4 The PREMO dataset

In this section, we present an overview of the dataset `PRemo` and also some preliminary analysis of its contents. The complete detailed dataset description is presented in our supplementary material [13].

4.1 Overview

By utilizing the methodology proposed in the previous section, we created a dataset, which is available as a JSON file on our website [13]. Alongside the dataset, the schema for the JSON file, and the scripts and tools utilized to build the dataset are also available.

Due to the careful design of our methodology, we provide a robust dataset that captures the communication dynamics among developers in PR discussions and effectively addresses the limitations observed in previous datasets (see Section 2). The dataset has 1,791 messages obtained from the 36 selected software projects (Table 1). Each message in the dataset contains:

(1) Software project name
(2) Message content
(3) URL to the message on GitHub
(4) Aggregate information for both labeling passes (with and without contextual information)
  (4.1) Aggregate sentiment polarity
  (4.2) Aggregate confidence (1-5, mean)
  (4.3) Agreement type (among everyone, only among software engineers, mixed or none)
(5) Anonymous information about how each evaluator labeled the message in each pass
  (5.1) Sentiment polarity
  (5.2) Sentiment intensity (1-4, separated between positive and negative for mixed messages)
  (5.3) Emotions identified (e.g., joy, anger, etc.)
  (5.4) Confidence level (1-5)

Figure 2 shows the distribution of the aggregate sentiments (i.e., the sentiment on which the evaluators agreed) in the dataset. Unless stated otherwise, all aggregates presented in this section are from the second pass, as they represent the results in which the evaluators evaluated each message in an informed manner (i.e., with contextual information). In summary, the dataset contains 521

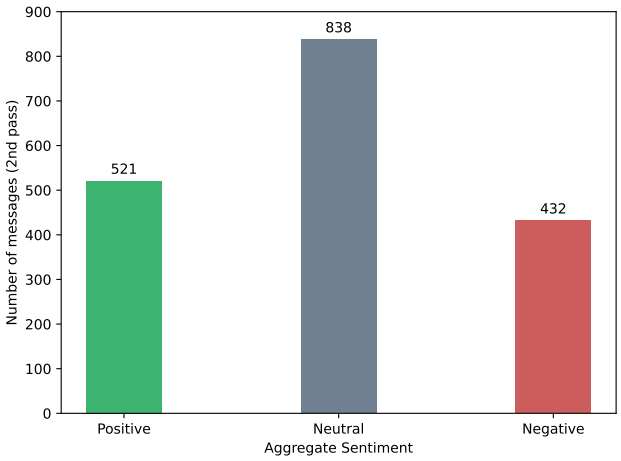

Figure 2: Dataset Distribution Between Different Sentiment Polarities (2nd pass)

(≈29%) messages labeled as positive, 432 (≈24%) as negative, and 838 (≈47%) as neutral. We were unable to achieve a balanced distribution; however, previous works have reasoned that a distribution of sentiments closer to the original data source can be preferable when compared to a balanced distribution [31].

Figure 3 displays the frequency of each emotion in our dataset according to Shaver's emotion model from psychology [33]. Instead of sentiment, where an aggregate was utilized based on the agreement between evaluators, we utilized the individual data each evaluator made to calculate this frequency. The emotions in the figure are grouped as having a positive (green) or negative (red) sentiment as defined by the emotion model [33]. The model [33] defines surprise as having a context-dependent polarity (i.e., it can be positive or negative depending on the context). Due to that, surprise appears twice in the graph's x-axis. Although some emotions are more predominant than others (i.e., sadness and joy), we believe we have achieved an acceptable number of examples for each type of emotion.

4.1.1 Agreement Between evaluators. To build PRemo, we utilized a triple validation process. This, combined with the fact that there are three sentiment classes (i.e., positive, negative, and neutral), means that disagreements can occur when aggregating the results. Figure 4 shows the frequency in which different levels of agreement occurred. We indicate cases where all evaluators reached consensus as *100% or full agreement*, cases where only two evaluators reached consensus as *66% or partial agreement*, and cases of unanimous disagreement among evaluators as *0% or no agreement*.

> The evaluators reached full or partial agreement in the majority of evaluated messages (≈97% of messages in the second pass).

A characteristic of our labeling process is that two groups of evaluators were involved in evaluating each message: software engineers and neuroscientists. We anticipated that there could be some disagreement between those two groups in terms of results. However, both groups agreed on ≈87% of messages.

4.1.2 Differences between the two passes. Another unusual characteristic of our labeling process was that the evaluator had to validate each message twice. In the first pass, they had to label a message while only considering its contents, while in the second pass, they had the entire context available (i.e., previous and following messages, the identities of people involved in the conversation, etc.). While the differences were not negligible, only ≈8% of messages had differences between the two evaluations done by each evaluator. The difference between the two passes was more present in the confidence levels (see next section).

> For the two passes, first considering only a single message and then the entire context available, no significant differences were found in evaluations conducted by each evaluator.

4.1.3 Evaluator Confidence. During the manual labeling process, each evaluator had to assign a confidence score (Very Little, Little, Moderate, Strong, and Very Strong Confidence) to their labels. These scores reflected the evaluators' confidence in the validity of their labeling. The mean confidence for the three evaluators, grouped by the agreement level per message and comparing each pass (the first or the second), can be observed in Figure 5.

> Overall, the confidence reported by the evaluators was positive, close to the Strong Confidence (4) mark for both passes (mean confidence: 1^{st} pass = 3.60, 2^{nd} pass = 3.95).

Notably, the figure shows a pattern associating the level of evaluator agreement and the confidence scores. For both passes, lower levels of agreement also had lower confidence. This could mean that some disagreements observed in the manual labeling process could have been caused by low confidence by the evaluators, which in turn has been reported by the evaluators to be caused by some messages containing challenging characteristics that made them difficult to evaluate. Examples of such messages include ambiguous tone, politeness masking emotion, multiple sentiments, sarcasm and irony.

4.2 Possible Uses

In this section, we discuss the potential applications of the constructed dataset PRemo, outlining its implications for research, practice, and tool development. These applications range from supporting automated tool creation to informing empirical studies and organizational decision-making. We also identify open challenges in the literature and opportunities for future research. We summarize our findings in Table 2. Most of the usages of PRemo mentioned in Table 2 and in the remainder of the paper remain speculative. However, they are backed by previous studies or gray literature that utilize similar datasets to achieve those goals.

The availability of fine-grained emotion annotations in PRemo opens up unique opportunities for both SE research and practice.

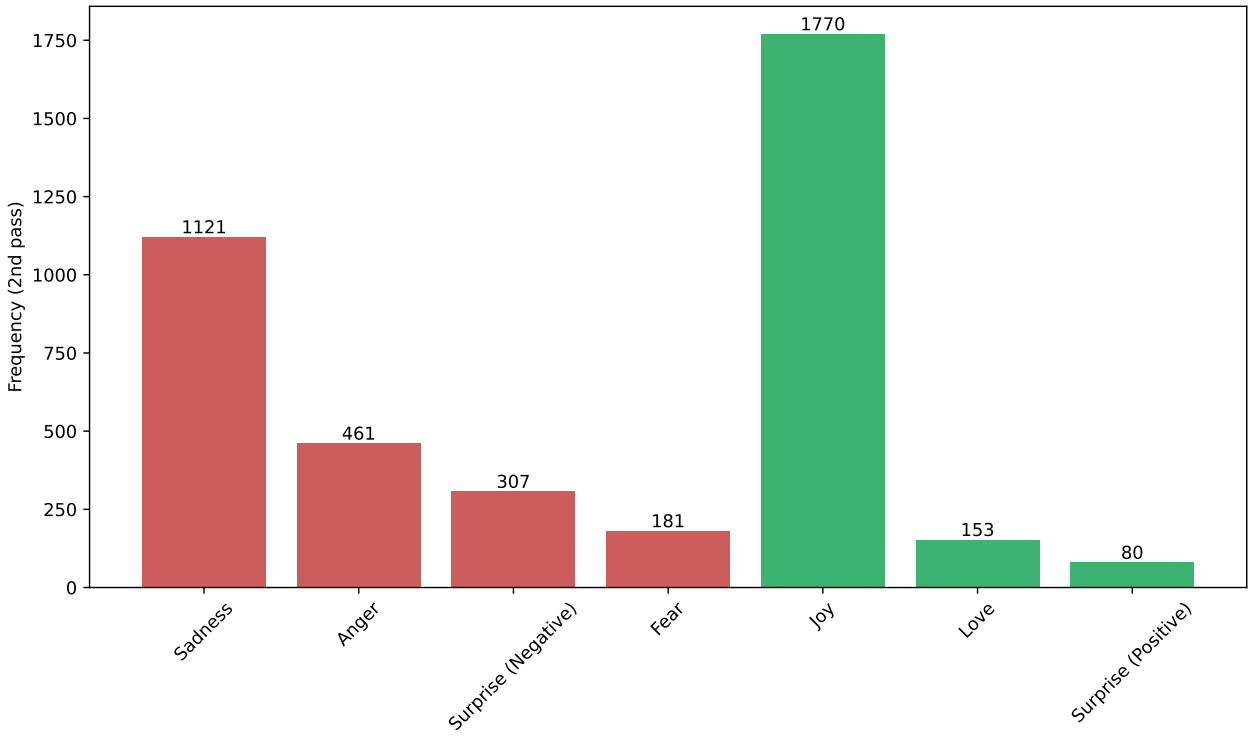

Figure 3: Aggregate Emotion Frequency in the Dataset (2nd pass)

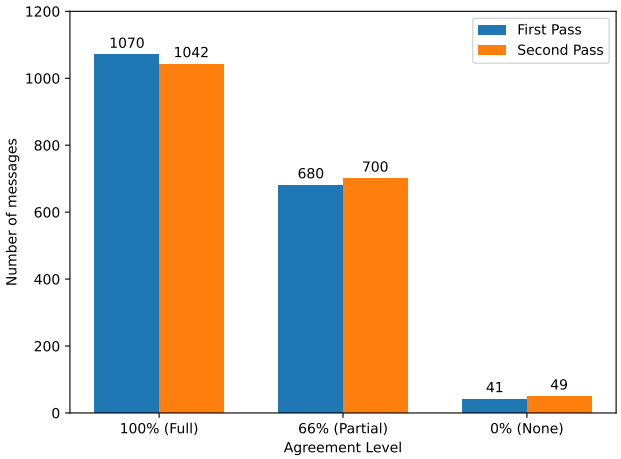

Figure 4: Frequency of Agreement Levels over Sentiment Reported Between the Evaluators in their Two Different Passes

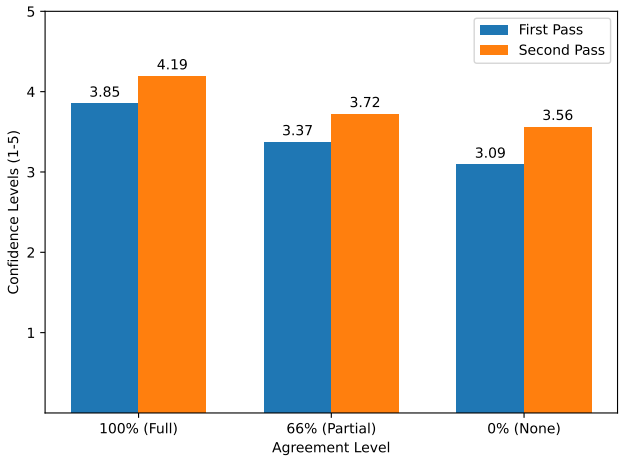

Figure 5: Confidence Levels Reported by the Evaluators in their Two Different Passes, Grouped by Agreement Level

In research, emotion-labeled data enables studies on how emotion dynamics influence development outcomes, such as productivity, collaboration quality, or design decision-making. For instance, prior work has shown that commits with negative sentiment are significantly more likely to introduce bugs [20], while positive (and some negative) emotional tones in developer chats were associated with increased productivity, measured in number of commits and lines of code [25]. Emotion trends can also be correlated with key events like refactorings, bugs, or architectural discussions, shedding light on the emotional undercurrents of technical debt accumulation or code review disagreements.

Table 2: Examples of Possible Uses of PRemo

Use Case	Description	Potential Benefits
Data acquisition	Our methodology enables researchers to extend PRemo by incorporating discussions from various sources.	Leads to the development of more robust and generalizable sentiment analysis tools.
Data preparation	Use of preprocessing, cleaning, feature engineering, and feature selection for preparing PRemo.	Reduces dimensionality, improves model accuracy and interpretability, and mitigates overfitting.
Learning algorithm	Use of PRemo to evaluate sentiment analysis learning approaches in software engineering.	Provides a standard for comparing state-of-the-art learning approaches.
Sentiment trends and dynamics over time	Use of sentiment and emotion data over time to monitor shifts in team morale and community tone.	Enables proactive responses to frustration and supports an empathetic development lifecycle.
Deployment of algorithms for practical use	Design of tools that automatically analyze comments, pull requests, and issues using PRemo.	Reduces manual moderation, improves communication, and fosters a healthier collaboration environment.
Real-world adoption	Evaluation of sentiment analysis utility and effectiveness in real organizational settings.	Enhances team collaboration and supports managers in identifying burnout or negative dynamics.
Transfer learning	Reuse of models trained on PRemo across other software development contexts.	Lowers the cost of adoption by reducing the need for task-specific model retraining.
Applications across SE domains	Use of emotion-level data to study affect in SE research, support managerial decisions, and improve affect-aware tools.	Enables deeper analysis of collaboration dynamics and tailored interventions based on specific emotions (e.g., fear vs. anger).

In organizational settings, emotion analysis can support engineering management in identifying periods of team frustration, confusion, or demotivation, which are not always evident through conventional metrics. For example, in-situ biometric studies during code reviews demonstrated that emotional states affect typing behavior and reviewer attention, revealing insights about when developers are struggling [41]. Similarly, experiments have linked specific emotional states to variations in developer performance on problem-solving tasks [5]. This can inform interventions to improve team well-being or communication practices. Emotion data can also improve sentiment-aware recommendation systems, such as adaptive bots for developers onboarding, or documentation assistants that react to affective cues. The granularity of emotional labeling—especially the presence of distinct emotions like sadness, anger, or fear—enhances these applications by allowing more precise responses than polarity-only models. For example, distinguishing frustration (anger) from uncertainty (fear) can guide whether a project manager should clarify goals or mediate interpersonal tension. As such, emotion-rich datasets such as PRemo serve as a foundational asset for advancing emotion-aware tooling and empirical understanding in modern SE contexts. Datasets such as the one presented in this paper can also be useful in a myriad of other contexts, which are discussed as follows.

Data acquisition. Our methodology can be used by other researchers to extend PRemo and thus enrich the dataset with additional discussions from other sources. Researchers can gather data from different repositories, platforms, or even from proprietary software development environments, thus broadening the scope and diversity of the dataset. Thus, encompassing a wider diversity of development practices, team dynamics, and communication styles, providing a more comprehensive understanding of sentiment in PR discussions across different contexts. Moreover, by incorporating data from diverse sources, researchers can investigate how sentiment analysis models perform in various software development environments, enabling the development of more robust and generalizable sentiment analysis tools. Overall, new data acquisition efforts present an opportunity to enrich PRemo and advance research in sentiment analysis in SE.

Data preparation. Preprocessing, cleaning, feature engineering, and feature selection are essential steps in preparing the dataset for sentiment analysis tasks. Preprocessing and cleaning the dataset involve transforming raw data into a suitable format for analysis by removing noise, irrelevant information, and inconsistencies (*e.g.*, handling special characters and URLs). In the context of PRemo, we have employed simple heuristics to clean the data as mentioned in Section 3. Further research could better automate this process. Furthermore, the raw data in PRemo may not be optimal for usage to build predictive models. Researchers can enhance the quality of PRemo in this use case by applying feature engineering and feature selection techniques. Feature engineering involves creating new features or transforming existing ones to better represent the underlying PR discussions and leading to more accurate and interpretable sentiment analysis models. Potential features could include word frequencies, number of comments, author reputation, or project activity level. Feature selection, on the other hand, involves choosing the most relevant subset of features to use in the model. Researchers could also explore sampling techniques to address any potential class imbalances. These can include resampling methods (such as

oversampling minority classes or undersampling majority classes). The choice of method would depend on the specific learning algorithm and the task at hand.

Learning algorithm. Observing numerous papers in the field reveals a prevalent focus on using state-of-the-art datasets, focusing mainly on building sentiment analysis tools and evaluating them in controlled environments. For instance, in a prior study [12], a subset of PRemo (specifically, the aggregate sentiment data for each message) served as a benchmark to assess the efficacy of state-of-the-art sentiment analysis tools in SE. While some tools perform acceptably, their performance is far from ideal, especially when classifying negative messages. Thus, a persistent question in the literature remains: *Which tools should practitioners employ, and under what conditions?* Notably, an open challenge is the absence of an actionable framework capable of automatically and *a priori* recommending an appropriate tool based on the given context. For example, the effectiveness of a tool may depend on the project characteristics, such as the project language, the work format (remote vs. onsite), or the team's background. Thus, further empirical research is needed to assist practitioners in systematically choosing suitable solutions.

Sentiment trends and dynamics over time. By analyzing the trends in sentiment over time, project teams could also gain insight into the general mood of their community (e.g., monitor shifts in team emotion or morale over release cycles), enabling them to proactively address potential sources of frustration. This approach not only streamlines communication within the development community but also paves the way for a more empathetic and responsive software development lifecycle.

Deployment of algorithms for practical use. Using PRemo robust tools can be built to analyze sentiment and emotions in GitHub messages and thus improve the workflow of developers. For example, a GitHub extension capable of automatically analyzing new comments, reduces the manual workload for project maintainers involved in moderation tasks. It could also assess new issues, PRs, and comments before developer submission, offering developers constructive feedback and tips to foster a positive community atmosphere. We believe that integrating tooling support is crucial for practical adoption, and the current lack may partly explain the gap between practice and research.

Real-world adoption in development organizations. Despite the advancements in sentiment analysis research, and the existence of anecdotal reports of the adoption of sentiment analysis in the industry, there remains a lack of concrete evidence regarding the extent and effects of its adoption in software development organizations. A critical transition from theoretical exploration to practical implementation involves moving beyond merely reporting evaluation metrics such as the accuracy of sentiment analysis models over open-source datasets. It requires a deeper examination of the applicability and effectiveness of sentiment analysis approaches within organizational contexts. Such studies include those driven by practical problems, such as how to support managers in fostering positive team dynamics and addressing mental health concerns among developers. To bridge this gap, future research endeavors should focus on deploying sentiment analysis algorithms onto real software systems and supporting managers in gaining a holistic understanding of team dynamics.

Transfer learning. A threat to the practical adoption of sentiment analysis lies in the high cost associated with learning-based approaches. Beyond the expenses incurred in training ML models, the cost of collecting data and building a large and robust dataset can be significant for many organizations. The need to train a completely new model from scratch for each usage scenario raises concerns. In this context, an open issue in the field is: *How sentiment analysis transfers to other contexts?* In this scenario, the reuse of learning-based models emerges as a promising solution. Previous tools [8] have followed in this path, by utilizing multiple datasets from different contexts to build a model that could possibly apply to a broader set of contexts. However, a recent evaluation has shown that it still does not perform ideally and more research is needed [12]. Thus, a research direction is to characterize which and to what extent factors influence the performance distribution of reusing existing models. The ultimate goal is to devise learning-based solutions that are effective, independent of the specific deployment context, thereby facilitating broader practical adoption.

5 Threats to Validity

This section discusses potential threats to the validity of this study and the steps taken to mitigate them. Next, we summarize the main threats to the validity concerning four groups proposed by Wohlin et al. [42]: *construct*, *internal*, *conclusion* and *external* validity.

Construct and Internal Validity: In the manual validation process, we faced two main challenges: (i) the selection and preprocessing of the PR messages; and (ii) the potential bias of the individuals conducting the validation. Concerning (i), using a classification tool to pre-categorize messages could influence the selection process. Such preprocessing might unintentionally bias our dataset towards emotions that the tool is more adept at identifying, possibly overlooking more subtle or complex emotional nuances that do not fit neatly into predefined categories. To mitigate this issue, we utilized a state-of-the-art tool (SentiStrengthSE) capable of detecting sentiment expression in messages. Although it is recognized that completely eliminating bias is unattainable, these measures substantially enhance the trustworthiness and quality of our dataset.

Regarding (ii), as in any qualitative analysis process, the inherent subjectivity of sentiment analysis poses a threat. To mitigate it, we designed a meticulous methodology to minimize subjectivity in manual labeling by involving SE and neuroscience evaluators and incorporated a triple validation model for each message. We also provided evaluators with comprehensive guidelines detailing the classification process and facilitating discussions to identify and standardize common patterns. Also, two pilot studies were conducted to fine-tune our documentation and methods, ensuring a more consistent and informed approach during the labeling phase.

Another acknowledged threat to construct validity is annotation fatigue. Due to the substantial volume of messages in our dataset, there is a risk that annotators may become mentally fatigued over time, potentially compromising the consistency and quality of the annotations, especially in the later stages of the process. To mitigate this risk, we instructed all validators to limit each annotation session

to a maximum of one hour. This time was informed by existing cognitive research suggesting that sustained attention and decision-making quality tend to decline after prolonged periods of focused activity. Additionally, validators were encouraged to take breaks between sessions and proceed at their own pace to maintain high standards of focus and reliability throughout the task.

Conclusion and External Validity: We acknowledge that the sample of 36 projects used here may not fully encapsulate the vast number of PR discussions happening across GitHub. We opted to work with a smaller sample of projects with criteria that enabled us to select a more valuable message selection, while still having a diverse set of projects. With this focus, we sought to build a dataset that offers valuable insights applicable across different SE contexts. This approach, combined with the thoughtful definition of our methodology, offers valuable credibility to the generalization of our dataset.

Another acknowledged threat to external validity concerns the language used in the analyzed PR messages. The vast majority of the messages were written in English, which may restrict the generalizability of our findings to open-source communities or development teams where communication predominantly occurs in other languages. Sentiment expression can vary significantly across languages, both in tone and in form, and cultural norms may influence how emotions are conveyed or interpreted in technical discussions. In future work, we plan to consider extending this analysis to multilingual repositories to better assess the robustness and applicability of sentiment analysis methods across diverse linguistic and cultural settings.

6 Conclusion and Future Work

This work represents a significant step forward in advancing emotion and sentiment analysis in the context of collaborative software development. We introduced the PRemo dataset, a novel resource specifically designed to capture both sentiment polarity and discrete emotions within PR discussions. By targeting this underexplored context, PRemo addresses gaps in existing datasets, which often focus solely on coarse-grained sentiment analysis and lack detailed affective annotations.

PRemo was constructed from a diverse sample of 36 active and mature open-source projects, ensuring broad representativeness across domains, languages, and development practices. To ensure high data quality and minimize subjectivity, we employed a rigorous methodology that combined triple validation, dual-pass labeling (with and without context), and the involvement of evaluators from both neuroscience and software engineering. In doing so, we not only produced a rich dataset with emotion-specific labels, intensity values, and confidence scores, but also established a reusable and transparent framework for future dataset curation efforts.

Beyond the dataset itself, we explored a wide range of potential applications for PRemo. These include benchmarking sentiment analysis tools, developing affect-aware software engineering assistants, studying affective trends over time, and deploying emotion-driven interventions in organizational settings. Our findings highlight the value of emotion-level data in enabling deeper insights, such as distinguishing between developer frustration, confusion, or appreciation—signals that are often blurred in traditional sentiment classification. We also emphasized real-world scenarios where such granularity can support engineering managers in monitoring team health or moderating toxic behavior in online communities.

PRemo has already been adopted in prior research [12] as a benchmark to evaluate sentiment classification tools in SE, demonstrating its immediate value to the community. Looking forward, we envision multiple directions for extending this work. Future studies may use PRemo to explore the generalizability of emotion models across software artifacts, improve transfer learning techniques for emotion classification, or design context-aware feedback tools that react to fine-grained emotional cues in developer communication. Additionally, future versions of the dataset could include PR discussions from private or industrial repositories, further bridging the gap between research and practice. Ultimately, we hope that the release of PRemo will catalyze future work on affective computing in software engineering.

ARTIFACT AVAILABILITY

The authors declare that the dataset and tools used to build the dataset are available at https://doi.org/10.5281/zenodo.17058478 [13]. They are also mirrored on GitHub at https://github.com/opus-research/sentiment-dataset/.

ACKNOWLEDGMENTS

This research was partially funded by the Brazilian funding agencies CAPES (88881.879016/2023-01), CAPES/Procad (175956), CAPES/PROEX, CNPq (434969/2018-4, 312149/2016-6, 141180/2021-8, GD), FAPESP (2023/00811-0), FAPERJ (200773/2019, 010002285/2019), NSF (2303042, 2247929, 2303612), and the Binational Cooperation Program CAPES/COFECUB (Ma1036/24). We also acknowledge the Brazilian company Stone[6] for their support.

Special thanks to all participants of the labeling process.

REFERENCES

[1] [n. d.]. 16 Popular Programming Languages and Their Uses Explained. https://mikkegoes.com/14-programming-languages-explained/. (Accessed on 03/28/2024).

[2] [n. d.]. Top Programming Languages and Their Uses - KDnuggets. https://www.kdnuggets.com/2021/05/top-programming-languages.html. (Accessed on 03/28/2024).

[3] [n. d.]. What are different programming languages used for? - FutureLearn. https://www.futurelearn.com/info/blog/what-are-different-programming-languages-used-for. (Accessed on 03/28/2024).

[4] 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)*. 106–111. https://doi.org/10.1109/ASE.2017.8115623

[5] Mohammed R. Anany, Heba Hussien, S. Aly, and Nourhan Sakr. 2019. Influence of Emotions on Software Developer Productivity. In *Proceedings of the 14th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*. 75–82. https://doi.org/10.5220/0008068800750082

[6] Caio Barbosa, Anderson Uchôa, Daniel Coutinho, Wesley KG Assunçao, Anderson Oliveira, Alessandro Garcia, Baldoino Fonseca, Matheus Rabelo, José Eric Coelho, Eryka Carvalho, et al. 2023. Beyond the Code: Investigating the Effects of Pull Request Conversations on Design Decay. In *2023 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 1–12.

[7] Caio Barbosa, Anderson Uchôa, Daniel Coutinho, Filipe Falcão, Hyago Brito, Guilherme Amaral, Vinicius Soares, Alessandro Garcia, Baldoino Fonseca, Marcio

[6] https://www.stone.com.br/

Ribeiro, et al. 2020. Revealing the social aspects of design decay: A retrospective study of pull requests. In *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*. 364–373.

[8] Fabio Calefato, Filippo Lanubile, Federico Maiorano, and Nicole Novielli. 2018. Sentiment Polarity Detection for Software Development. In *Proceedings of the 40th International Conference on Software Engineering* (Gothenburg, Sweden) *(ICSE '18)*. Association for Computing Machinery, New York, NY, USA, 128. https://doi.org/10.1145/3180155.3182519

[9] Fabio Calefato, Filippo Lanubile, Federico Maiorano, and Nicole Novielli. 2018. Sentiment polarity detection for software development. In *Proceedings of the 40th International Conference on Software Engineering*. 128–128.

[10] Fabio Calefato, Filippo Lanubile, and Nicole Novielli. 2017. Emotxt: a toolkit for emotion recognition from text. In *2017 seventh international conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW)*. IEEE, 79–80.

[11] Zhenpeng Chen, Yanbin Cao, Huihan Yao, Xuan Lu, Xin Peng, Hong Mei, and Xuanzhe Liu. 2021. Emoji-powered sentiment and emotion detection from software developers' communication data. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 30, 2 (2021), 1–48.

[12] Daniel Coutinho, Luisa Cito, Maria Vitória Lima, Beatriz Arantes, Juliana Alves Pereira, Johny Arriel, João Godinho, Vinicius Martins, Paulo Vítor CF Libório, Leonardo Leite, et al. 2024. " Looks Good To Me;-)": Assessing Sentiment Analysis Tools for Pull Request Discussions. In *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering*. 211–221.

[13] Daniel Coutinho, Juliana Alves Pereira, Breno Braga Neves, João Correia, Caio Barbosa, Wesley K.G. Assunção, Igor Steinmacher, Marco Gerosa, Augusto Baffa, and Alessandro Garcia. 2025. *opus-research/sentiment-dataset: CBSOFT 2025 Artifacts Festival - Version v2*. https://doi.org/10.5281/zenodo.17058478

[14] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. 2012. Social coding in GitHub: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*. ACM, 1277–1286.

[15] Rafael de Mello, Roberto Oliveira, Leonardo Sousa, and Alessandro Garcia. 2017. Towards effective teams for the identification of code smells. In *2017 IEEE/ACM 10th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, 62–65.

[16] Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. 2020. GoEmotions: A dataset of fine-grained emotions. *arXiv preprint arXiv:2005.00547* (2020).

[17] Paul Ekman, Tim Dalgleish, and M Power. 1999. Basic emotions. *San Francisco, USA* (1999).

[18] Mateus Freira, Josemar Caetano, Johnatan Oliveira, and Humberto Marques-Neto. 2018. Analyzing the impact of feedback in GitHub on the software developer's mood. In *2018 International Conference on Software Engineering & Knowledge Engineering*.

[19] Daniel Graziotin, Xiaofeng Wang, and Pekka Abrahamsson. 2013. Are happy developers more productive? The correlation of affective states of software developers and their self-assessed productivity. In *Product-Focused Software Process Improvement: 14th International Conference, PROFES 2013, Paphos, Cyprus, June 12-14, 2013. Proceedings 14*. Springer, 50–64.

[20] Syed Fatiul Huq, Ali Zafar Sadiq, and K. Sakib. 2020. Is Developer Sentiment Related to Software Bugs: An Exploratory Study on GitHub Commits. In *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 527–531. https://doi.org/10.1109/SANER48275.2020.9054801

[21] Md Rakibul Islam and Minhaz F. Zibran. 2017. Leveraging Automated Sentiment Analysis in Software Engineering. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. 203–214. https://doi.org/10.1109/MSR.2017.9

[22] Md Rakibul Islam and Minhaz F. Zibran. 2018. DEVA: Sensing Emotions in the Valence Arousal Space in Software Engineering Text. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing* (Pau, France) *(SAC '18)*. Association for Computing Machinery, New York, NY, USA, 1536–1543. https://doi.org/10.1145/3167132.3167296

[23] Robbert Jongeling, Subhajit Datta, and Alexander Serebrenik. 2015. Choosing your weapons: On sentiment analysis tools for software engineering research. In *2015 IEEE international conference on software maintenance and evolution (ICSME)*. IEEE, 531–535.

[24] Chris F Kemerer and Mark C Paulk. 2009. The impact of design and code reviews on software quality: An empirical study based on psp data. *IEEE Trans. Softw. Eng. (TSE)* 35, 4 (2009), 534–550.

[25] Miikka Kuutila, M. Mäntylä, and Maëlick Claes. 2020. Chat activity is a better predictor than chat sentiment on software developers productivity. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*. https://doi.org/10.1145/3387940.3392224

[26] Bin Lin, Fiorella Zampetti, Gabriele Bavota, Massimiliano Di Penta, Michele Lanza, and Rocco Oliveto. 2018. Sentiment Analysis for Software Engineering: How Far Can We Go?. In *Proceedings of the 40th International Conference on Software Engineering* (Gothenburg, Sweden) *(ICSE '18)*. Association for Computing Machinery, New York, NY, USA, 94–104. https://doi.org/10.1145/3180155.3180195

[27] Nikolaos Lykousas, Constantinos Patsakis, Andreas Kaltenbrunner, and Vicenç Gómez. 2019. Sharing emotions at scale: The vent dataset. In *Proceedings of the International AAAI Conference on Web and Social Media*, Vol. 13. 611–619.

[28] 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. *Emp. Softw. Eng. (ESE)* 21, 5 (2016), 2146–2189.

[29] Alessandro Murgia, Parastou Tourani, Bram Adams, and Marco Ortu. 2014. Do Developers Feel Emotions? An Exploratory Analysis of Emotions in Software Artifacts. In *Proceedings of the 11th Working Conference on Mining Software Repositories* (Hyderabad, India) *(MSR 2014)*. Association for Computing Machinery, New York, NY, USA, 262–271. https://doi.org/10.1145/2597073.2597086

[30] Nicole Novielli, Fabio Calefato, Davide Dongiovanni, Daniela Girardi, and Filippo Lanubile. 2020. Can We Use SE-Specific Sentiment Analysis Tools in a Cross-Platform Setting?. In *Proceedings of the 17th International Conference on Mining Software Repositories* (Seoul, Republic of Korea) *(MSR '20)*. Association for Computing Machinery, New York, NY, USA, 158–168. https://doi.org/10.1145/3379597.3387446

[31] Martin Obaidi, Lukas Nagel, Alexander Specht, and Jil Klünder. 2022. Sentiment analysis tools in software engineering: A systematic mapping study. *Information and Software Technology* (2022), 107018.

[32] Marco Ortu, Alessandro Murgia, Giuseppe Destefanis, Parastou Tourani, Roberto Tonelli, Michele Marchesi, and Bram Adams. 2016. The Emotional Side of Software Developers in JIRA. In *Proceedings of the 13th International Conference on Mining Software Repositories* (Austin, Texas) *(MSR '16)*. Association for Computing Machinery, New York, NY, USA, 480–483. https://doi.org/10.1145/2901739.2903505

[33] Phillip Shaver, Judith Schwartz, Donald Kirson, and Cary O'connor. 1987. Emotion knowledge: further exploration of a prototype approach. *Journal of personality and social psychology* 52, 6 (1987), 1061.

[34] Mauricio Soto, Zack Coker, and Claire Le Goues. 2017. Analyzing the impact of social attributes on commit integration success. In *Mining Software Repositories (MSR), 2017 IEEE/ACM 14th International Conference on*. IEEE, 483–486.

[35] Patanamon Thongtanunam, Shane McIntosh, Ahmed E Hassan, and Hajimu Iida. 2015. Investigating code review practices in defective files: An empirical study of the qt system. In *12th MSR*. IEEE Press, 168–179.

[36] Jason Tsay, Laura Dabbish, and James Herbsleb. 2014. Influence of social and technical factors for evaluating contribution in GitHub. In *Proceedings of the 36th international conference on Software engineering*. 356–366.

[37] Jason Tsay, Laura Dabbish, and James Herbsleb. 2014. Let's Talk about It: Evaluating Contributions through Discussion in GitHub. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering* (Hong Kong, China) *(FSE 2014)*. Association for Computing Machinery, New York, NY, USA, 144–154. https://doi.org/10.1145/2635868.2635882

[38] Anderson Uchôa, Caio Barbosa, Daniel Coutinho, Willian Oizumi, Wesley KG Assunçao, Silvia Regina Vergilio, Juliana Alves Pereira, Anderson Oliveira, and Alessandro Garcia. 2021. Predicting design impactful changes in modern code review: A large-scale empirical study. In *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*. IEEE, 471–482.

[39] Anderson Uchôa, Caio Barbosa, Willian Oizumi, Publio Blenílio, Rafael Lima, Alessandro Garcia, and Carla Bezerra. 2020. How does modern code review impact software design degradation? an in-depth empirical study. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 511–522.

[40] Bogdan Vasilescu, Stef Van Schuylenburg, Jules Wulms, Alexander Serebrenik, and Mark GJ van den Brand. 2014. Continuous integration in a social-coding world: Empirical evidence from GitHub. In *Software Maintenance and Evolution (ICSME), 2014 IEEE International Conference on*. IEEE, 401–405.

[41] Hana Vrzakova, Andrew Begel, Laura Mehtätalo, and Roman Bednarik. 2020. Affect Recognition in Code Review: An In-situ Biometric Study of Reviewer's Affect. *J. Syst. Softw.* 159 (2020). https://doi.org/10.1016/j.jss.2019.110434

[42] C Wohlin, P Runeson, M Host, MC Ohlsson, B Regnell, and A Wesslen. 2000. Experimentation in software engineering: an introduction.

[43] Yue Yu, Huaimin Wang, Vladimir Filkov, Premkumar Devanbu, and Bogdan Vasilescu. 2015. Wait for it: Determinants of pull request evaluation latency on GitHub. In *Mining software repositories (MSR), 2015 IEEE/ACM 12th working conference on*. IEEE, 367–371.