“Looks Good To Me ;-)": Assessing Sentiment Analysis Tools for Pull Request Discussions

Daniel Coutinho
Luisa Cito
Maria Vitória Lima
Beatriz Arantes
Juliana Alves Pereira
Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Rio de Janeiro
Brazil

Johny Arriel
João Godinho
Vinicius Martins
Paulo Libório
Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Rio de Janeiro
Brazil

Leonardo Leite
Federal University of Alagoas (UFAL), Maceió
Brazil

Alessandro Garcia
Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Rio de Janeiro
Brazil

Wesley K. G. Assunção
North Carolina State University (NCSU), Raleigh
USA

Igor Steinmacher
Northern Arizona University (NAU), Flagstaff
USA

Augusto Baffa
Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Rio de Janeiro
Brazil

Baldoino Fonseca
Federal University of Alagoas (UFAL), Maceió
Brazil

ABSTRACT

Modern software development relies on cloud-based collaborative platforms (*e.g.*, GitHub and GitLab). In these platforms, developers often employ a pull-based development approach, proposing changes via pull requests and engaging in communication via asynchronous message exchanges. Since communication is key for software development, studies have linked different types of sentiments embedded in the communication to their effects on software projects, such as bug-inducing commits or the non-acceptance of pull requests. In this context, sentiment analysis tools are paramount to detect the sentiment of developers' messages and prevent potentially harmful impact. Unfortunately, existing state-of-the-art tools vary in terms of the nature of their data collection and labeling processes. Yet, there is no comprehensive study comparing the performance and generalizability of existing tools utilizing a dataset that was designed and systematically curated to this end, and in this specific context. Therefore, in this study, we design a methodology to assess the effectiveness of existing sentiment analysis tools in the context of pull request discussions. For that, we created a dataset that contains ≈1.8K manually labeled messages from 36 software projects. The messages were labeled by 19 experts (neuroscientists and software engineers), using a novel and systematic manual classification process designed to reduce subjectivity. By applying these existing tools to the dataset, we observed that while some tools]perform acceptably, their performance is far from ideal, especially when classifying negative messages. This is interesting since negative sentiment is often related to a critical or unfavorable opinion. We also observed that some messages have characteristics that can make them harder to classify, causing disagreements between the experts and possible misclassifications by the tools, requiring more attention from researchers. Our contributions include valuable resources to pave the way to develop robust and mature sentiment analysis tools that capture/anticipate potential problems during software development.

EASE 2024, June 18–21, 2024, Salerno, Italy
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1701-7/24/06
https://doi.org/10.1145/3661167.3661189

CCS CONCEPTS

• **Software and its engineering → Collaboration in software development**; • **Human-centered computing → Collaborative and social computing**.

KEYWORDS

repository mining, human aspects, sentiment analysis

ACM Reference Format:
Daniel Coutinho, Luisa Cito, Maria Vitória Lima, Beatriz Arantes, Juliana Alves Pereira, Johny Arriel, João Godinho, Vinicius Martins, Paulo Libório, Leonardo Leite, Alessandro Garcia, Wesley K. G. Assunção, Igor Steinmacher, Augusto Baffa, and Baldoino Fonseca. 2024. “Looks Good To Me ;-)": Assessing Sentiment Analysis Tools for Pull Request Discussions. In *28th International Conference on Evaluation and Assessment in Software Engineering (EASE 2024), June 18–21, 2024, Salerno, Italy.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3661167.3661189

1 INTRODUCTION

Cloud-based collaborative software development platforms (*e.g.*, GitHub and GitLab) are ubiquitous in the modern landscape of software development [6]. Utilizing these platforms, software development is mostly done in a remote, asynchronous, and distributed way [35]. These platforms implement an approach called pull-based development [11, 12], in which developers propose changes to the software system by creating what GitHub calls *pull requests*.[1] Within these pull requests, developers can engage with each other in two ways: (i) pull request discussions and (ii) code review comments. While the former is commonly used to revise higher-level concepts (e.g., design decisions [4] and discussions about non-functional requirements [31]), the latter tends to focus on specific excerpts of the source code submitted in the pull request [37, 41].

Pull-based development promotes the communication among developers [23, 33]. Studies in the literature have shown that many aspects related to communication (*e.g.*, size of the discussion and discussion topics) can influence the codebase in different ways, such as accelerating design decay [3]. In recent studies, researchers have found that *Sentiment*[2] is another aspect of communication that has been seen to affect the quality of commits, requiring future fixes [14] and influencing the rejection and/or acceptance of pull requests [32]. In this context, sentiment analysis tools are used to detect the sentiment of developers' messages and prevent negative impact. This has led to the creation of a wide variety of software-engineering-specific sentiment analysis tools [28].

Despite contributing to developing the field of sentiment analysis in modern software development, existing tools vary in terms of how robustly they were built [30]. The main limitations of the existing tools are associated with (i) the nature of the data/discussions used to train the tools and (ii) the arbitrary labeling process that mostly lacks a robust methodology. In terms of the *nature of the data/discussions*, these sentiment analysis tools use texts from different collaborative development platforms, such as bug or issue tracking platforms (*e.g.*, Bugzilla [8] and JIRA [16]), or Q&A websites (*e.g.*, Stack Overflow [29]). However, these bug or task tracking platforms utilize a more decentralized approach, where discussions about bug reports or task assignments are done through platforms that are external to version control. This affects how and what the developers of a project discuss. Conversely, platforms such as GitHub and GitLab utilize a centralized approach, where these facets of development are centralized, motivating active discussion among developers [42]. Regarding the *labeling process*, most sentiment analysis tools do not define emotions using pre-established models accepted by the literature (*e.g.*, Shaver *et al.*[38] and Ekman [10]), as they either do not specify it or use an ad-hoc definition. Few existing tools are based on an emotion model from psychology [5].

The goal of our work is to overcome the two limitations described above to enable a robust assessment of existing state-of-the-art software-engineering-specific sentiment analysis tools. For that, we designed a methodology to assess the performance of these tools by utilizing a novel dataset as a benchmark, which we called PRemo [1]. Then, PRemo is used to assess five state-of-the-art tools.

Experimental results show that while some tools perform acceptably in the context of pull request discussions, their results were not ideal, especially when classifying negative messages. This is interesting since negative sentiment is usually related to a critical or unfavorable opinion. Thus, existing tools are misclassifying relevant information. Interestingly, these results differ from previous studies on the effectiveness of these tools in detecting negative messages [28]. We also observed messages that were harder to classify, causing disagreement among experts about the existence or polarity of the sentiments. Thus, we provide hints about the characteristics of these contentious messages that can impact the overall performance of the classification of negative messages.

The contributions of our work are manyfold. We propose a methodology to classify messages in a robust and systematic way by experts from different areas of expertise (neuroscience and software engineering, in our case). Additionally, we relied on an established emotion model from the psychology literature [38]. We create PRemo, a curated open-source dataset composed of 1,791 manually labeled pull request messages. These messages were mined from 36 software systems. All of our artifacts are publicly available in our replication package [1]. Our results support understanding how effective state-of-the-art tools are for sentiment analysis on the pull-based software development model. Practitioners can rely on our insights to use specific tools that align with their needs. Researchers can also use the methodology proposed for PRemo to build other datasets tailored to other contexts. Moreover, our findings (e.g., the characterization of messages that were harder to classify) and dataset can be used by tool builders to create new or improve existing tools.

2 BACKGROUND

2.1 Sentiment Analysis and Emotion Models

The use of digital platforms has led to a predominance of human interaction via online means and, consequently, has generated an extremely large amount of textual data. These data retain valuable information and have broadened the scope of research on day-to-day communication. Advances in *Natural Language Processing* (NLP) techniques have received significant attention due to their effectiveness in language modeling [24]. In software engineering, another factor contributing to its notoriety is the possibility of applying these models to SE-specific data, where their output can provide significant knowledge about workspace dynamics and productivity [19, 21].

Machine Learning (ML) techniques are often paired with NLP to build and improve language-related models. In this context, sentiment analysis has driven advances within the scope of sentence classification. Its versatility is presented through a number of wildly differently structured levels of sentence analysis. The technique is tailored for the subjective assessment of sentence polarities, classifying them as being positive, negative, or neutral. These categories should represent the author's general judgment towards the sentence [24].

To evaluate a sentence's polarity, it is crucial to determine strict criteria to ensure the replicability and flexibility of the classification. For this purpose, in ML and lexicon-based approaches, classifiers/methods such as feature extraction and bag of words are utilized. Similarly, for manual annotation, a theoretical emotion model provides a conceptual framework that guides the labeling process and the construction of an accurate and reliable classifier.

[1]https://docs.github.com/pull-requests
[2]We refer to *Sentiment* as the subject of *Sentiment Analysis*, which, for this work, is defined as the process of extracting the subjectivities of a sentence (i.e., an opinion or emotion, which can be positive, negative, or neutral) [7].

The ML model selects the relevant features, and the theoretical emotion model allows the interpretation of the results under a solid conceptual basis.

Among a variety of emotion models, Shaver's [38] has been widely used in the software development context for manual annotation [20, 25]. Table 1 was adapted from [20] and used as a guideline for our classification (see Section 3). It depicts an adaptation of Shaver's model [38] of emotions and establishes the possible polarities of each emotion. Its high granularity accounts for a continuous spectrum of emotions while also determining a categorical set of those. Shaver's model adopts a hierarchical approach to emotions, viewing those as "fuzzy" entities. Therefore, emotions are a set of categories that present within vague boundaries. As such, emotions exist within a continuum of evaluation (positive to negative polarity), potency (weak to strong), and activity (high to low). Although emotions do not present sharp boundaries between each other, the Shaver model proposes a set of basic entities that best seem to encompass the spectrum of emotional mental representations [38].

The Shaver model establishes six basic emotions: love, anger, joy, sadness, fear, and surprise. These emotions can be classified within positive polarity (love, joy, and sometimes surprise) and negative polarity (anger, sadness, fear, and sometimes surprise). Each of these emotion categories contains a set of sub-clusters, classified as second-level emotions by [20].

Table 1: Emotion Model Utilized in This Work

Polarity	Basic Emotions	Second-Level Emotions
Positive	Love	Affection Lust Longing
	Joy	Cheerfulness Zest Contentment Optimism Pride Enthrallment
Negative	Anger	Irritation Exasperation Rage Disgust Envy Torment
	Sadness	Suffering Sadness Disappointment Shame Neglect Sympathy
	Fear	Horror Nervousness
Context dependent	Surprise	Surprise

2.2 SE-Based Sentiment Analysis Tools

SE-Based Sentiment Analysis Tools have been developed in response to the observed inadequacy of traditional sentiment analysis tools when applied to SE datasets. Conventional tools are designed primarily for social media texts or product reviews. These tools were found to be poorly suited for SE contexts. Recognizing the expressive nature of developers during various SE tasks, the authors took on the challenge of creating a tool tailored to this niche.

SE-Based Sentiment Analysis Tools employ the categorization of sentences in accordance with an emotion model, such as Shaver's model [20], while others used an ad-hoc approach [2, 9, 20]. There are many studies conducted in the literature to validate, test, and/or compare Sentiment Analysis Tools [2, 9, 20, 28]. These studies present several limitations. (1) Often, they do not present a theoretical emotion model to guide sentiment analysis, following what is called an ad-hoc approach. This indicates that the criteria for labeling messages are formulated according to the validators. Novielli et al. [28] show that SE-specific customization may not guarantee reasonable accuracy if ad-hoc annotation is performed. (2) Existing approaches do not analyze the entire context of the exchanged messages, as the focus lies on the sentence itself. When the context is included, it is related only to sentences prior to the one aimed at classification [25]. However, isolating a sentence from its context can increase the chance of misinterpreting the sender's intention. (3) The method of selecting sentences to perform sentiment analysis may impact the polarity outcome in manual labeling.

It is worth noting that a recent study in the field [28] shows that SE-specific sentiment analysis tools performed well in identifying positive and negative sentiment in issue discussions, but had lower accuracy in identifying neutral sentiment. Notice that neutral sentiment varies among authors: some authors [5, 25] call "neutral" messages that have more than one emotion, while others identify it as the absence of emotions in a given message (see Section 3).

We employed Obaidi et al.'s [30] mapping to identify SE-based Sentiment Analysis Tools. We then eliminated tools that were not available and chose five tools to be used in our study. We are also aware that newer techniques, such as transformers [44], have been used for sentiment analysis in the domain of software engineering. Although these techniques are effective and are a part of the state-of-the-art, we opted to focus on the tools that are still the most popular, at least according to the aforementioned recent literature review [30]. Therefore, we present tools chosen as the subject of analysis for this paper, presenting their main characteristics as reported by their authors, as follows.

SentiStrengthSE [16] tool is a version developed based on SentiStrength [39, 40]. While the first was trained with non-technical data from social media (*e.g.* Twitter posts, forum posts, movie reviews), the second was trained and is intended for sentiment analysis in the SE context. To develop the tool, the authors initially carried out a qualitative and quantitative study of SentiStrength based on a dataset with 5,992 issue comments [34] manually labeled by experts from JIRA issue tracking platform.[3] SentiStrengthSE uses a lexical approach, similar to that used in SentiStrength. The authors present the precision, recall, and F-score values for both tools, showing that SentiStrentghSE outperforms its original version in all aspects. However, it presented a few limitations. SentiStrentghSE

[3]https://www.atlassian.com/software/jira/service-management

presented difficulties in identifying proper nouns, such as the developer's name "Harsh". It failed to identify "Harsh" and considered the word negative, leading to a negative analysis of the message.

Senti4SD [5] is a sentiment polarity classifier designed specifically for Q&A websites, such as Stack Overflow, GitHub and JIRA discussions. Senti4SD incorporates lexicon-based, keyword-based, and semantic features tailored to the SE domain. The tool's semantic features are derived from a Distributional Semantic Model (DSM) [5] created using word2vec. Compared to existing tools like SentiStrength, Senti4SD offers a more accurate classification, particularly in reducing the mislabeling of neutral and positive posts as negative. This enhanced accuracy is particularly evident when considering specific datasets used for the tool's development and validation. The first dataset comprises 4,423 Stack Overflow posts, including questions, answers, and comments, all of which were manually annotated with polarity labels by twelve experts. The second dataset includes 5,869 JIRA sentences manually annotated by three experts. Additionally, the third dataset encompasses over 7,000 sentences from GitHub pull requests and commits. [28].

SentiCR [2] is a sentiment analysis tool that has been customized for code review interactions in SE. It uses NLTK universal parts-of-speech (POS) tagger and chunking to identify negation phrases and improve the accuracy of sentiment analysis. Three experts used an ad-hoc approach to manually label 1,600 review comments from Gerrit.[4] They used two classes, negative and non-negative, as a final distribution. This dataset was then leveraged to evaluate the effectiveness of seven well-known sentiment analysis tools (Afinn [27], NLTK [13], SentiStrength [40], TextBlog [22], USent [36], NLTK Vader [15], Vivekn [26]), highlighting the necessity and efficacy of a domain-specific solution like SentiCR. Furthermore, the tool has been applied to other SE datasets, such as GitHub issue comments and Stack Overflow posts, to improve the quality of software development.

DEVA (Detecting Emotions in Valence) [17] is a tool for automated sentiment analysis in SE context. A differential of DEVA when compared to other tools, is that along the valence analysis (positive or negative), there is the arousal analysis (high or low). With this approach, it is possible to implement a bi-dimensional model, adding more depth to the sentiment classification. It is worth noticing the difference between valence and polarity. While polarity is in a hierarchical classification model, valence is used alongside arousal, both have positive or negative values attributed to them. Sentiments are classified as excitement, stress, depression, relaxation, or neutral. To capture both valence and arousal, they used two different lexical dictionaries, one for each dimension of analysis. The tool was created based on a dataset containing 1,795 issue comments in JIRA, with a precision of 82.19% [17]. Those messages were manually labeled by three computer science graduate students. However, although DEVA presents itself as empirically superior to its baseline [17], there is a lack of efficiency due to the struggle to detect emotions in texts with more subtle emotions, such as sarcasm and irony. Another important limitation is the interpretation of emoticons. Rather than prioritize the context where an emoticon is inserted, DEVA tends to evaluate what the emoticon represents in the dictionary and ignore the rest of the sentence, which can lead to a misleading analysis.

Conclusion. Except for SentiCR and Senti4SD, which utilize a dataset of code review messages from pull requests, most of the previously mentioned tools leverage data from contexts that are different from pull request discussions. Those differences in context can create differences in language, which might harm the performance of these sentiment analysis tools. Furthermore, only Senti4SD utilizes a robust methodology for its labeling process. The other tools use an ad-hoc methodology, which might introduce bias in their results, especially in a highly subjective task such as sentiment analysis. Therefore, the need to evaluate these tools in the context of pull request discussions becomes evident. To do that, we designed and built PRemo, a curated open-source dataset composed of 1,791 manually labeled pull request messages mined from 36 software systems hosted on GitHub.

3 STUDY DESIGN

3.1 Goal and Research Questions

Our was defined using the Goal Question Metric (GQM) approach [43], as follows: **Characterize** the performance of state-of-the-art sentiment analysis tools in the specific environment of modern software development with cloud-based collaborative platforms **for the purpose of** assessing the effectiveness of existing tools **from the viewpoint of** researchers and practitioners **in the context of** a systematically curated dataset composed of GitHub pull request discussions. Thus, we tackle the following research questions (RQs).

[**RQ$_1$:** *How do state-of-the-art sentiment analysis tools perform when classifying messages in the context of pull request discussions?*

Rationale: Pull request discussions are key to software development on GitHub, since they are used by developers to discuss topics related to the project, such as code changes and design decisions [11]. While investigating the state-of-the-art SE-specific sentiment analysis tools, we identified two main limitations that indicate that they might not be suited for usage in this context, namely the (i) nature/source of their data and (ii) the robustness of their labeling process. To address these limitations and advance the knowledge of existing support for sentiment analysis, this RQ aims to systematically investigate the performance of existing tools.
Method: To answer RQ$_1$, we designed a methodology to create a dataset, which is systematically composed and curated by experts from different contexts (i.e., neuroscientists and software engineers), addressing the subjectiveness and lack of robustness of existing studies. Therefore, we propose PRemo, a robust open-source dataset of 1,791 manually labeled pull request messages mined from 36 software systems hosted on GitHub. This dataset is used to assess five state-of-the-art SE-specific sentiment analysis tools (see Section 2), in order to evaluate their performance.
Metrics: To analyze performance, we utilize the well-known metrics of precision, recall, and F-score. Based on these metrics, we present a quantitative analysis comparing the performance of five state-of-the-art sentiment analysis tools in the context of pull request discussions.

[4]https://www.gerritcodereview.com/

[**RQ$_2$:** *Does (dis)agreement among experts about the existence or polarity of sentiment have any influence on the performance of sentiment analysis tools?*

Rationale: To build PRemo, we executed a manual classification process to label messages from pull request discussions (see Section 3.2). During this process, multiple experts (i.e., neuroscientists and software engineers) evaluated each message. We observed that while they were able to reach an agreement on their classification for most messages, this was not always the case. Those experts raised key points that made some messages more challenging to classify. Thus, we wanted to investigate how existing sentiment analysis tools would perform for messages having different levels of difficulty to classify, even for experts. Thus, we relied on information about the (dis)agreement among experts to investigate their influence on the performance of sentiment analysis tools.
Method: For this RQ, we combine the performance data from the previous RQ with information about the agreement among experts that manually classified each message, which is available in PRemo, to investigate the relationship between those two pieces of information. Also, during the manual classification process, experts were instructed to make observations about patterns that made messages more challenging to classify. These observations are also used for the analysis to answer this RQ.
Metrics: For RQ$_2$, we aim to investigate the relationship between the performance (*i.e.*, precision, recall, and F-score metrics) for each sentiment analysis tool and the agreement level between experts when manually labeling each of the messages. In addition to quantitative analysis, we also report qualitative analysis, as we highlight and discuss multiple examples, that were curated by the experts, of problematic characteristics they observed in some of the messages that were more challenging to classify.

3.2 Study Steps and Procedures

To answer the two RQs of our study, we first define a methodology to (i) build a robust dataset that captures sentiments in the context of pull request discussions and (ii) execute five state-of-the-art sentiment analysis tools. Our methodology is composed of six steps, which are described in detail below.

3.2.1 Project Selection. Firstly, to build the dataset, we first selected open-source projects from which we collected pull request discussions. We used the following inclusion criteria: the open-source project has to (i) be hosted on GitHub; (ii) use a pull-based development workflow; (iii) have at least 1k commits and pull requests; (iv) be at least 5 years old; and (v) be in active development. Additionally, we only selected projects implemented in Python, Javascript, Typescript, or Java. These programming languages were chosen due to them being often employed in different domains.[5]

Based on the aforementioned criteria, we selected 36 projects. The full list of projects selected is available in our replication package [1]. While more projects fulfilled our criteria, we intentionally selected a limited number of projects to maintain a good balance between the number of messages selected per project and domain diversity. Domain diversity is relevant when selecting each software project, to guarantee the robustness of our dataset.

3.2.2 Selecting messages From pull request discussions. While mining data from GitHub, messages are usually collected from three main places: issues, pull request discussions, and code review comments (also a part of pull requests). Issues have no specific subject and, therefore, can be extremely unstructured. Also, they cannot always be linked to source code, which can impose limits on the investigation of their effects. Differently, pull requests always have associated changes to the source code. In the context of pull requests, we did not utilize code review comments as they can be too specific, since they mostly focus only on proposing changes to a few lines on the associated code change at a time. Thus, in this work, we focus on pull request discussions, where developers still discuss the code changes (therefore, there is a central topic to the discussion), but usually focus on higher-level aspects, such as broader design decisions [11].

After selecting the projects, we used the GitHub API to collect all messages exchanged in pull request discussions throughout the history of each of the projects selected. During this collection, we performed a filtering process, /colorblueutilizing bot-specific information provided by the GitHub API and by the username (detecting if it includes “BOT”), to exclude messages sent by bots. This process resulted in the collection of approximately one million messages. The next step was to select messages to be manually labeled by experts. For that, since technical messages are often neutral [18], we employed a pre-classification step utilizing the SentiStrengthSE tool [16] (see Section 2). SentiStrengthSE classified the messages into three groups, namely positive, negative, and neutral. To validate our selection, both in terms of suitability of the methodology and distribution of the data, the final selection was done in two rounds. For the *first round*, 1,000 messages from 11 Java projects were randomly selected (*i.e.*, 350 random positive messages, 350 random negative messages, and 300 random neutral messages). For the *second round*, after manually labeling the first round of messages (see Step 3), additional messages were selected from the 25 remaining Java, Python, Javascript, and Typescript projects. For each project, we selected 48 messages (*i.e.*, 20 random positive messages, 20 random negative messages, and eight random neutral messages), totaling 1,200 messages. The values for each class are unbalanced, as we identified during our first round of messages that the distribution of our dataset was heavily skewed towards neutral messages.

3.2.3 Manually labeling messages. After we finished selecting a sample of messages, we started the sentiment analysis with a manual labeling process. This process was performed by 19 experts. Among them, 16 were software engineering students (ranging from undergraduate to PhD students), with varying levels of experience, and three students were neuroscience undergraduate students who were familiar with the theoretical emotion model utilized in the labeling process. While not all the experts had experience performing research, we opted for this group, and called them experts as all of them, even the neuroscience students, have had previous experience with the language utilized in our context (*i.e.*, either analyzing or having practical experience with how GitHub users communicate).

[5]What are Programming Languages & What are They Used For? https://www.linkedin.com/pulse/what-programming-languages-used-fhg-consultants/

To guarantee that the experts follow the procedures defined, the classification process was executed utilizing a self-developed tool. Before initiating the classification, each of the experts had to read a guide that was available in our tool. The guide explained how they should consider the emotion model in their classifications, and also provided several examples of pre-classified messages, facilitating the onboarding process.

The labeling process was designed to use a triple validation model, where each of the messages was always classified by one of the neuroscience students and two software engineering students. This ensured that all messages were classified using the same standard.The labeling process for each message was also done in two rounds. In the *first round*, the experts only had access to an individual message. Then, in the *second round*, they had access to the entire PR discussion containing the message they were classifying.

To reduce subjectivity in the manual classification, we based the classification on the theoretical model proposed by Shaver *et al.* [38] (see Section 2). This model pairs different emotions with a polarity: positive or negative. However, neutral labels have been utilized in previous research utilizing similar models [5, 25]. Thus, we also included a neutral label, which encompasses the absence of emotion and cases where the difference in intensity of positive and negative emotions are zero (*i.e.*, their intensity is the same). In summary, in our final classification, messages can be classified as having *positive*, *negative*, and *neutral* polarities. Since each message was evaluated by three experts, we considered the majority decision as the final polarity for a message. For each answer, the experts also had to report a confidence level. For that, we utilized a 5-point scale.

3.2.4 Discussing the disagreements. During Step 3, we identified a few messages where the three experts designated to a message disagreed regarding their classification. This represented a total of ≈3% (54) out of the 2,200 messages. In these cases, we asked the experts to discuss the message and reach an agreement. The final label, which was agreed as the result of those discussions, was finally included in the dataset.

3.2.5 Excluding messages from the dataset. While we took some actions to remove invalid messages from our sample in Step 2 (*e.g.*, filtering messages from bots), we identified several additional messages that had to be removed from the dataset during the manual classification. Therefore, to clean our dataset, we employed the following criteria to exclude messages from the dataset: (i) messages with no textual content (*e.g.*, only user mentions, links, etc. Emojis were considered textual content); (ii) messages written by bots; (iii) messages not in English; and (iv) duplicated messages.

3.2.6 Executing State-of-the-art Sentiment Analysis Tools. Finally, we selected five state-of-the-art sentiment analysis tools: SentiStrength [40], SentiStrengthSE [16], DEVA [17], SentiCR [2], and Senti4SD [5]. Details for each of those tools are available in Section 2. We decided to focus on software-engineering-specific tools, except for the first one (SentiStrength), which was included as a baseline for general-purpose tools.

To generate the results and answer our two RQs, each of those tools was evaluated by running them against the pull request discussion messages contained in our dataset. Then, we collected the metrics described as part of the RQs.

4 RESULTS AND DISCUSSION

4.1 The PRemo dataset

One contribution resulting from our work is the PRemo dataset [1], obtained by executing the procedures described in Section 3. PRemo contains 1,791 messages from 36 software projects that were manually classified with the help of 19 experts. PRemo avoids the limitations mentioned in the previous sections, as its data is sourced directly from pull request discussions, and it leverages a robust labeling process that is based on previous literature (see Section 3.2). Regarding the classes of messages, PRemo contains 521 (≈29%) messages labeled as positive, 432 (≈24%) labeled as negative, and 838 (≈47%) labeled as neutral. This could be a good characteristic for future research using this dataset, as previous works have argued that, when training sentiment analysis classifiers, a more natural distribution might be preferable instead of a perfectly balanced dataset [30].

Also, our manual labeling used a triple validation process with three experts; thus, it was expected that they would disagree on some messages. Based on that, we grouped the classified messages into three agreement levels: (i) *total agreement*, in which all experts agreed; (ii) *partial agreement*, in which two out of three experts agreed; and (iii) *no agreement*, where all experts disagreed. This amounts to, respectively, 1042 (≈58%), 700 (≈39%), and 49 (≈3%) out of the 1791 messages in our dataset.

By utilizing this dataset, in the next sections, we answer our RQs.

4.2 Performance of State-of-the-art Sentiment Analysis Tools on the PRemo dataset (RQ$_1$)

To answer RQ$_1$, we first executed the state-of-the-art sentiment analysis tools against our dataset, following the Step 6 of our methodology (see Section 3). Table 2 presents the performance precision, recall, and F1-score for the five tools. The results are separated by class, since four out of the five tools are multi-class classifiers. The table also includes the average performance for all classes to illustrate the overall performance of each tool. We analyze and discuss the results presented in the table in what follows.

Performance Requirements. First, we assess the performance of the tools focusing on F-score, given that we consider that precision and recall are both important in this scenario, and should be prioritized based on a case-by-case basis. As a baseline for minimum performance, we decided to utilize the percentage of messages with a unanimous agreement among all experts during manual labeling (for more information about the expert agreement, see Section 4.3). In other words, we expect the tools to correctly classify the messages for at least the ones that all experts agreed about the sentiment. Based on that, we consider 0.58 (1,042 out of 1,791 messages = 58%) as the minimum acceptable for precision, recall, and F-score for the tools. Nonetheless, it is important to note that this value represents the minimum acceptable performance, and it is far from being a good result in practical settings.

Table 2: Per Class Performance of State-of-the-art Sentiment Analysis Tools

Class	Tool	Precision	Recall	F1-score
Positive	SentiStrength	55.32%	78.89%	0.65
	SentiStrengthSE	**79.23%**	**62.96%**	**0.70**
	DEVA	66.49%	72.36%	0.69
	Senti4SD	56.47%	71.21%	0.63
Negative	SentiStrength	45.06%	69.68%	0.55
	SentiStrengthSE	43.56%	79.86%	0.56
	DEVA	**48.41%**	**70.37%**	**0.57**
	Senti4SD	57.32%	42.59%	0.49
	SentiCR	42.79%	39.81%	0.41
Neutral	SentiStrength	80.53%	36.52%	0.50
	SentiStrengthSE	79.32%	55.37%	0.65
	DEVA	**79.87%**	**56.8%**	**0.66**
	Senti4SD	65.81%	63.84%	0.65
Macro Avg.	SentiStrength	60.30%	61.70%	0.61
	SentiStrengthSE	**67.37%**	**66.06%**	**0.67**
	DEVA	59.87%	59.21%	0.60
	Senti4SD	64.92%	66.51%	0.66
	SentiCR	42.79%	39.81%	0.41

Average Performance. We can observe in Table 2, that, on average, four of the five state-of-the-art sentiment analysis tools reach the minimum baseline of performance we established: SentiStrength, SentiStrengthSE, DEVA, and Senti4SD. Nevertheless, SentiStrengthSE and Senti4SD stand out, as they have the best performance (0.67 and 0.66 F-score, respectively). However, even the best-performing tools are still far from having ideal performance, and may not be performant enough to be used in real applications or to be used as sources of data for future research.

SentiCR stood out as the worst performer. At first glance, this result is surprising, since this tool also utilizes data from GitHub pull requests. However, they specifically utilize code review comments, which are messages that directly reference source code, and are usually shorter and more technical. Therefore, the messages PRemo are very different from those in SentiCR's ground truth.

Finding 1: On average, most sentiment analysis tools reach our minimum expected performance, except for SentiCR. Specifically, SentiStrengthSE and Senti4SD stand out as having the best performance. However, even the best-performing tools are far from having ideal performance. SentiCR, despite being the tool that is the most closely related to pull request discussions, stood out as the tool with the worst performance.

Per-Class Performance. Since four out of five tools analyzed are multiclass classifiers (SentiCR is a binary classifier for negative messages), it is important to individually analyze their performance in terms of each of the classes they classify.

The per-class performance analysis shows a different scenario from what the average shows. While the performance for classifying neutral messages is very close to the average, we can observe that the average actually hides the more extreme results for the positive and negative classes, which are opposite to each other. The results for classifying positive messages are good, with SentiStrength actually reaching the 0.7 F-score mark, with DEVA also coming very close to reaching that value. Since the former performs better in terms of precision and the latter in terms of recall, both could be used in applications, depending on whether the specific use case prioritizes having fewer false positives or having more true positives (fewer false negatives). On the other hand, the performance for classifying negative messages was very poor. While SentiStrength, SentiStregthSE, and DEVA were close, none of the tools reached our minimum expected performance. This result contradicts previous studies, which observed neutral messages as being the worst performers in software-engineering-specific sentiment analysis tools [20].

Finding 2: When looking at the classification of positive messages, the analyzed tools performed much better than the overall results. SentiStrengthSE and DEVA stood out as having good performers, with the former having better precision and the latter having higher recall. Differently, the performance for classifying negative messages in the context of pull request discussions was very poor, with none of the tools achieving acceptable performance.

This finding can be explained by some of the characteristics that our experts noted when working on the manual labeling. They observed that positive messages often had specific words or expressions that denoted positivity. This could possibly also be one of the reasons for the good performance of SentiStrengthSE, since it utilizes a lexicon/dictionary approach. They also observed that, unlike positive messages, negative messages were frequently very subtle, sometimes even employing irony or sarcasm, which has been noted by previous works as a problem for these tools [30].

4.3 Relationship Between Expert Agreement and the Performance of the Tools (RQ$_2$)

For Step 7 of our study (see Section 3.2), we executed a process in which the experts discussed the messages where all developers disagreed, in order to reach an agreement. During that discussion, the experts noted how subtle those "no agreement" messages were, unlike other messages (*i.e.*, messages with other agreement levels), which they found more intuitive. Due to that, multiple experts stated they would change their original label if possible. Seeing those results, we decided to investigate whether those messages, which are more challenging to classify, would also be more likely to be classified wrongly by the sentiment analysis tools.

Table 3 shows the average performance of the sentiment analysis tools analyzed when classifying messages with different agreement levels. In this table, we can observe that there is a clear difference in performance for different agreement levels, which suggests that some characteristics of the messages that caused disagreements between experts also affect the tools. While the differences between the group "no agreement" and the others may be inconclusive, given the small sample size (49 messages), even the difference between the "total" and "partial agreement" groups is significant.

Analyzing the messages in which all the experts agreed ("total agreement"), two of the tools analyzed performed very well:

Table 3: Average Performance of State-of-the-art Sentiment Analysis Tools per Agreement Level

Agreement Level	Tool	Precision	Recall	F1-score
Total	SentiStrength	68.59%	70.73%	0.70
	SentiStrengthSE	75.99%	76.59%	0.76
	DEVA	69.01%	67.24%	0.68
	Senti4SD	**75.47%**	**78.24%**	**0.77**
	SentiCR	47.59%	40.09%	0.44
Partial	SentiStrength	48.48%	50.30%	0.49
	SentiStrengthSE	**56.42%**	**52.84%**	**0.55**
	DEVA	48.31%	48.43%	0.48
	Senti4SD	50.90%	51.66%	0.51
	SentiCR	39.61%	40.00%	0.40
None	SentiStrength	44.44%	33.02%	0.38
	SentiStrengthSE	35.85%	31.17%	0.33
	DEVA	41.09%	40.75%	0.41
	Senti4SD	**44.39%**	**41.89%**	**0.43**
	SentiCR	12.50%	20.00%	0.15

SentiStrengthSE and Senti4SD, reaching F-scores of 0.76 and 0.77, respectively. SentiStrength and DEVA also performed well, with 0.70 and 0.68 F-scores, respectively. In the case of the messages that had a “partial agreement” (two out of three experts agreed), we can see a high reduction in performance, with a drop of 0.22 in the F-score when comparing top performers with the full agreement group. We can also see another drop of 0.12 when looking at the messages where the experts disagreed. This implies that there are characteristics to the messages in these two agreement-level groups that might be hampering the work of both the human experts and the sentiment analysis tools.

Finding 3: There is a significant reduction in the performance of the sentiment analysis tools when experts disagree about the existence or polarity of sentiment in a pull request message.

Similarly to RQ$_1$, since most of the sentiment analysis tools investigated are multiclass classifiers, we analyzed the performance of the tools on a class-by-class basis. Table 4 presents the performance of the tools separated by agreement level, but this time groups the results by class.

Analyzing the table, we can notice that, when comparing the performance for different classes, the overall trends look pretty similar. In the “total” and “partial agreement” groups, the trend is that tools have their best performance for classifying positive messages, and their worst performance when classifying negative messages remains. However, there are some differences. For positive messages in the “total agreement” group, we can see that all four tools performed well, with SentiStrengthSE and DEVA reaching an F-score of above 0.8. Also, for the first time, we can see a case where the tools had good performance when classifying negative messages. In fact, we can observe a similar tradeoff as the one that is seen in Finding 2. We have two tools that perform well, namely DEVA and Senti4SD. The former performs better overall, thanks to a very good recall of ≈85%, while the latter has close to 70% precision.

Table 4: State-of-the-art Sentiment Analysis Tools Performance, Grouped by Agreement Level

Agreement Level	Class	Tool	Precision	Recall	F1-score
Total	Positive (331)	SentiStrength	64.29%	87.01%	0.74
		SentiStrengthSE	87.54%	74.32%	0.80
		DEVA	**78.19%**	**83.38%**	**0.81**
		Senti4SD	64.10%	80.36%	0.71
	Negative (222)	SentiStrength	50.57%	80.18%	0.62
		SentiStrengthSE	48.10%	91.44%	0.63
		DEVA	**56.12%**	**84.68%**	**0.68**
		Senti4SD	69.46%	52.25%	0.60
		SentiCR	47.59%	40.09%	0.44
	Neutral (489)	SentiStrength	90.91%	44.99%	0.60
		SentiStrengthSE	92.33%	64.01%	0.76
		DEVA	**92.09%**	**66.67%**	**0.77**
		Senti4SD	73.48%	69.12%	0.71
Partial	Positive (177)	SentiStrength	43.66%	66.10%	0.53
		SentiStrengthSE	**68.64%**	**45.76%**	**0.55**
		DEVA	50.52%	55.37%	0.53
		Senti4SD	44.55%	55.37%	0.49
	Negative (205)	SentiStrength	40.60%	59.02%	0.48
		SentiStrengthSE	**40.29%**	**67.80%**	**0.51**
		DEVA	41.33%	54.63%	0.47
		Senti4SD	45.89%	32.68%	0.38
		SentiCR	39.61%	40.00%	0.40
	Neutral (318)	SentiStrength	61.19%	25.79%	0.36
		SentiStrengthSE	60.34%	44.97%	0.52
		DEVA	60.85%	44.97%	0.52
		Senti4SD	**54.49%**	**57.23%**	**0.56**
None	Positive (13)	SentiStrength	22.22%	46.15%	0.30
		SentiStrengthSE	6.67%	7.69%	0.07
		DEVA	15.00%	23.08%	0.18
		Senti4SD	**31.82%**	**53.85%**	**0.40**
	Negative (5)	SentiStrength	11.11%	40.00%	0.17
		SentiStrengthSE	12.00%	60.00%	0.20
		DEVA	**18.18%**	**80.00%**	**0.30**
		Senti4SD	12.50%	20.00%	0.15
		SentiCR	12.50%	20.00%	0.15
	Neutral (31)	SentiStrength	100.00%	12.90%	0.23
		SentiStrengthSE	88.89%	25.81%	0.40
		DEVA	100.00%	22.58%	0.37
		Senti4SD	**78.95%**	**48.39%**	**0.60**

In the case of neutral messages, while the performance in terms of F-score here is not very different from the average performance from Table 2, the precision values are higher, with three out of five tools reaching above 90%.

Finding 4: When considering only messages with “total agreement”, we can see for the first time a situation where some of the tools (DEVA and Senti4SD) perform well when classifying negative messages.

We see major differences for the group “partial agreement”. We see a large reduction in the overall performance, with the best F-score for all combinations of classes and tools being 0.56. Also, while positive messages are still more precisely classified (mostly because of SentiStrenghSE), in terms of F-score, positive and neutral messages are very close. Also, for the messages with “no agreement”, we see another large reduction in performance, except for Senti4SD, which was an outlier in terms of performance when classifying neutral messages. However, it is not possible to draw any deeper conclusions for this group due to the small sample size.

The observed reduction in performance was reflected in the confidence values reported during the manual labeling process. We measured that by using a 5-point scale, and the average for each agreement level was: ≈3.85 for "total agreement", ≈3.40 for "partial agreement", and ≈3.25 for "no agreement". We can see that those values indeed decreased as agreement also decreased.

While we have no conclusive evidence for why this difference in performance of the tools between agreement levels occurs, those differences are in line with the previously stated observations made by the experts who participated in the manual labeling. To illustrate these observations, we present three examples from messages in which experts fully disagreed, with characteristics that made their manual labeling challenging. These characteristics could also possibly lead sentiment analysis tools to misclassify messages. These messages are:

Mixed polarity. Some messages, especially those with mixed polarity, were noted as especially challenging by experts. For example, the message "*Thanks for the proposal. Unfortunately, ignoring the exception isn't the right thing to do*"[6] can be interpreted as expressing weak positive and negative emotions. Some experts might believe that one of the two polarities is stronger, while others might consider the message neutral.

Requires additional context. Some messages require additional information, that is not contained in the message itself, to be accurately classified. For example, the message "*I signed it!*".[7] At first glance, it might seem that the author of the message is expressing either joy (positive) or irritation (negative) at someone. In reality, this message was a standardized command, issued to a bot. We also observed cases where a long wait time between messages caused some animosity to arise in the discussion, which was not clear when looking only at the message. Finally, another type of additional context that helped the experts in their manual labeling was the contents of previous/following messages. Our dataset was designed in a way in which we can, in the future, investigate the influence of this last type of additional context.

Routine messages, formality, and emoticons/emojis. Another occurrence noted by the experts was that in routine messages (*e.g.*, "*LGTM*", and "*thanks for the contribution*") and/or very formal messages, some common expressions or the addition of emotions/emojis can indicate an emotion where there is no one. Conversely, sentiment analysis tools that do not support emojis/emotions might overlook messages where these special characters do, indeed, indicate emotions. For example, the excerpt "*I thought I mention it at least ;-)*"[8] was added by the author at the end of a more technical message. Some experts thought this emoticon expressed an emotion, while others did not.

[6]https://github.com/spring-projects/spring-boot/pull/27542#issuecomment-895119391
[7]https://github.com/google/gson/pull/1229#issuecomment-357974195
[8]https://github.com/spring-projects/spring-boot/pull/18961#issuecomment-552851662

5 THREATS TO VALIDITY

We discuss threats to the study validity [43] as follows.

Construct and Internal Validity: Regarding the construction of our dataset, we observed that experts got more confidence in their answers as they gained experience throughout the process. Since the manual classification was executed over multiple rounds, this might have affected the quality of early classifications. To mitigate this threat, we provided comprehensive documentation to experts, describing how the classification should be performed. We also had frequent meetings to discuss common patterns. Moreover, we executed two pilot studies to polish our documentation and procedures before beginning the original labeling process. Finally, while the use of an emotion model that is established from previous literature improves the robustness of our labeling process, it does not guarantee that no bias was introduced in our dataset. However, it increases the reliability of the results.

For the assessment of the state-of-the-art tools, the conditions in which they were executed may have affected their results. Messages from pull request discussions are unstructured, and often include different forms of noise (*e.g.,* links, mentions, quotes, code blocks, emojis, or non-ascii characters). To mitigate this issue, we executed a comprehensive pre-processing step on the messages to remove those types of noises. However, some forms of noise may have remained in some messages. We also utilized a one-size-fits-all approach to our pre-processing. While this ensures fairness in terms that all tools are classifying the same message, some tools deal better with different forms of noise than others, so we might have inadvertently removed advantages that specific tools might present over others.

Conclusion and External Validity: Since our dataset represents data from 36 software projects, results based on our dataset are not generalizable, as the dataset is not representative of the full context of pull request discussions on GitHub. In order to mitigate this threat, we designed several steps of our methodology (*i.e.,* our project and message selection criteria) to increase the variety of projects and messages included in the manual labeling process (described in Section 3).

6 CONCLUSION

In this work, we have investigated the state-of-the-art sentiment analysis tools, focusing on how they perform in the context of GitHub pull request discussions. Unfortunately, the current state-of-the-art sentiment analysis tools for GitHub pull request discussions "do not look good :-(".[9] Our results indicate that the performance of the sentiment analysis tools analyzed is far from ideal. Some tools perform acceptably in certain situations. For example, while SentiStrengthSE is the overall best performer (with F-score equal to 0.67), it favors classifying positive messages with high precision (≈79%). In this use case (classifying positive messages), DEVA also performs similarly and favors recall (≈72%). Our results also hinted at a relationship between expert disagreement and a decrease in performance.

Besides our findings, we also provide the following contributions: (i) a novel and robust methodology to manually classify messages

[9]Reference to the title of this paper.

while avoiding subjectivity; and (ii) an open-source dataset containing ≈1.8k manually labeled messages, which is available as part of our replication package [1]. We anticipate that the contributions and findings detailed in the paper can help researchers and practitioners understand the effectiveness of the state-of-the-art sentiment analysis tools when applied to pull request discussions. We also expect that our dataset can be used to build newer, more powerful, sentiment analysis tools. In future work, we intend to continue utilizing PRemo to explore novel research questions, and also explore the usage of more advanced ML techniques, such as transformers and LLMs, alongside our dataset, to improve the state-of-the-art in software-engineering-specific sentiment analysis.

DATA AVAILABILITY

The data and scripts for this study are openly available at [1].

ACKNOWLEDGMENTS

Special thanks to Breno Braga, Camila Terra, Joanne Carneiro, Júlia Azevedo, Liege Fontenele, Luana Bueno, and Theo Canuto for participating in the manual labeling.

This work is partially funded by FAPERJ/PDR-10 (202073/2020), CAPES (88881.879016/2023-01), FAPERJ (200.510/2023, 211.033/2019, 202.621/2019), CNPq (315711/2020-5, 141276/2020-7, 141054/2019-0, 141180/2021-8), CAPES/Procad (175956), CAPES/Proex (88887.373933 / 2019-00), and IEEA-RJ (001/2021).

REFERENCES

[1] 2023. Replication Package. https://github.com/opus-research/sentiment-replication/. https://doi.org/10.5281/zenodo.11118201

[2] Toufique Ahmed, Amiangshu Bosu, Anindya Iqbal, and Shahram Rahimi. 2017. SentiCR: A customized sentiment analysis tool for code review interactions. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 106–111. https://doi.org/10.1109/ASE.2017.8115623

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

[4] João Brunet, Gail C Murphy, Ricardo Terra, Jorge Figueiredo, and Dalton Serey. 2014. Do developers discuss design?. In *Proceedings of the 11th Working Conference on Mining Software Repositories*. 340–343.

[5] 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

[6] Jürgen Cito, Gerald Schermann, John Erik Wittern, Philipp Leitner, Sali Zumberi, and Harald C. Gall. 2017. An Empirical Analysis of the Docker Container Ecosystem on GitHub. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. 323–333. https://doi.org/10.1109/MSR.2017.67

[7] Nhan Cach Dang, María N Moreno-García, and Fernando De la Prieta. 2020. Sentiment analysis based on deep learning: A comparative study. *Electronics* 9, 3 (2020), 483.

[8] Jin Ding, Hailong Sun, Xu Wang, and Xudong Liu. 2018. Entity-level sentiment analysis of issue comments. In *Proceedings of the 3rd International Workshop on Emotion Awareness in Software Engineering*. 7–13.

[9] Jin Ding, Hailong Sun, Xu Wang, and Xudong Liu. 2018. Entity-Level Sentiment Analysis of Issue Comments. In *Proceedings of the 3rd International Workshop on Emotion Awareness in Software Engineering* (Gothenburg, Sweden) *(SEmotion '18)*. Association for Computing Machinery, New York, NY, USA, 7–13. https://doi.org/10.1145/3194932.3194935

[10] Paul Ekman. 1992. An argument for basic emotions. *Cognition and Emotion* 6, 3-4 (1992), 169–200. https://doi.org/10.1080/02699939208411068 arXiv:https://doi.org/10.1080/02699939208411068

[11] Georgios Gousios, Martin Pinzger, and Arie van Deursen. 2014. An exploratory study of the pull-based software development model. In *Proceedings of the 36th international conference on software engineering*. 345–355.

[12] Georgios Gousios, Andy Zaidman, Margaret-Anne Storey, and Arie Van Deursen. 2015. Work practices and challenges in pull-based development: the integrator's perspective. In *37th ICSE*, Vol. 1. 358–368.

[13] Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. 168–177.

[14] Syed Fatiul Huq, Ali Zafar Sadiq, and Kazi Sakib. 2019. Understanding the effect of developer sentiment on fix-inducing changes: An exploratory study on github pull requests. In *2019 26th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 514–521.

[15] Clayton Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the international AAAI conference on web and social media*, Vol. 8. 216–225.

[16] 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

[17] 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

[18] 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.

[19] Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. 2023. Natural language processing: State of the art, current trends and challenges. *Multimedia tools and applications* 82, 3 (2023), 3713–3744.

[20] 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

[21] Saskia Locke, Anthony Bashall, Sarah Al-Adely, John Moore, Anthony Wilson, and Gareth B Kitchen. 2021. Natural language processing in medicine: a review. *Trends in Anaesthesia and Critical Care* 38 (2021), 4–9.

[22] Steven Loria et al. 2018. textblob Documentation. *Release 0.15* 2, 8 (2018), 269.

[23] Ian R McChesney and Seamus Gallagher. 2004. Communication and co-ordination practices in software engineering projects. *Information and Software Technology* 46, 7 (2004), 473–489.

[24] Kostadin Mishev, Ana Gjorgjevikj, Irena Vodenska, Lubomir T. Chitkushev, and Dimitar Trajanov. 2020. Evaluation of Sentiment Analysis in Finance: From Lexicons to Transformers. *IEEE Access* 8 (2020), 131662–131682. https://doi.org/10.1109/ACCESS.2020.3009626

[25] 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

[26] Vivek Narayanan, Ishan Arora, and Arjun Bhatia. 2013. Fast and accurate sentiment classification using an enhanced Naive Bayes model. In *Intelligent Data Engineering and Automated Learning–IDEAL 2013: 14th International Conference, IDEAL 2013, Hefei, China, October 20-23, 2013. Proceedings 14*. Springer, 194–201.

[27] Finn Årup Nielsen. 2011. A new ANEW: Evaluation of a word list for sentiment analysis in microblogs. *arXiv preprint arXiv:1103.2903* (2011).

[28] 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

[29] Nicole Novielli, Fabio Calefato, and Filippo Lanubile. 2018. A gold standard for emotion annotation in stack overflow. In *Proceedings of the 15th international conference on mining software repositories*. 14–17.

[30] 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.

[31] Anderson Oliveira, João Correia, Leonardo Sousa, Wesley KG Assunção, Daniel Coutinho, Alessandro Garcia, Willian Oizumi, Caio Barbosa, Anderson Uchôa, and Juliana Alves Pereira. 2023. Don't Forget the Exception!: Considering Robustness Changes to Identify Design Problems. In *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. IEEE, 417–429.

[32] Marco Ortu, Giuseppe Destefanis, Daniel Graziotin, Michele Marchesi, and Roberto Tonelli. 2020. How do you propose your code changes? Empirical analysis of affect metrics of pull requests on github. *IEEE access* 8 (2020), 110897–110907.

[33] Marco Ortu, Tracy Hall, Michele Marchesi, Roberto Tonelli, David Bowes, and Giuseppe Destefanis. 2018. Mining communication patterns in software development: A github analysis. In *Proceedings of the 14th international conference on predictive models and data analytics in software engineering*. 70–79.

[34] 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

[35] María J Palazzi, Jordi Cabot, Javier Luis Canovas Izquierdo, Albert Solé-Ribalta, and Javier Borge-Holthoefer. 2019. Online division of labour: emergent structures in open source software. *Scientific reports* 9, 1 (2019), 13890.

[36] Nikolaos Pappas, Georgios Katsimpras, and Efstathios Stamatatos. 2013. Distinguishing the popularity between topics: a system for up-to-date opinion retrieval and mining in the web. In *Computational Linguistics and Intelligent Text Processing: 14th International Conference, CICLing 2013, Samos, Greece, March 24-30, 2013, Proceedings, Part II 14*. Springer, 197–209.

[37] Caitlin Sadowski, Emma Söderberg, Luke Church, Michal Sipko, and Alberto Bacchelli. 2018. Modern code review: a case study at google. In *Proceedings of the 40th international conference on software engineering: Software engineering in practice*. 181–190.

[38] 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.

[39] Mike Thelwall, Kevan Buckley, and Georgios Paltoglou. 2012. Sentiment strength detection for the social web. *Journal of the American Society for Information Science and Technology* 63, 1 (2012), 163–173. https://doi.org/10.1002/asi.21662 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/asi.21662

[40] Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. 2010. Sentiment strength detection in short informal text. *Journal of the American society for information science and technology* 61, 12 (2010), 2544–2558.

[41] 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.

[42] Giovanni Viviani, Michalis Famelis, Xin Xia, Calahan Janik-Jones, and Gail C Murphy. 2019. Locating latent design information in developer discussions: A study on pull requests. *IEEE Transactions on Software Engineering* 47, 7 (2019), 1402–1413.

[43] Claes Wohlin, Per Runeson, Martin Höst, Magnus Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in Software Engineering* (1st ed.). Springer Science & Business Media.

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