Understanding the Low Inter-Rater Agreement on Aggressiveness on the Linux Kernel Mailing List

Thomas Bock[a], Niklas Schneider[a], Angelika Schmid[b], Sven Apel[a] and Janet Siegmund[c]

[a]Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
[b]IBM, Nürnberg, Germany
[c]University of Technology Chemnitz, Chemnitz, Germany

ARTICLE INFO

Keywords:
software developer communication
sentiment analysis
human annotation

ABSTRACT

Communication among software developers plays an essential role in open-source software (OSS) projects. Not unexpectedly, previous studies have shown that the conversational tone and, in particular, aggressiveness influence the participation of developers in OSS projects. Therefore, we aimed at studying aggressive communication behavior on the Linux Kernel Mailing List (LKML), which is known for aggressive e-mails of some of its contributors. To that aim, we attempted to assess the extent of aggressiveness of 720 e-mails from the LKML with a human annotation study, involving multiple annotators, to select a suitable sentiment analysis tool.

The results of our annotation study revealed that there is substantial disagreement, even among humans, which uncovers a deeper methodological challenge of studying aggressiveness in the software-engineering domain. Adjusting our focus, we dug deeper and investigated why the agreement among humans is generally low, based on manual investigations of ambiguously rated e-mails. Our results illustrate that human perception is individual and context dependent, especially when it comes to technical content. Thus, when identifying aggressiveness in software-engineering texts, it is not sufficient to rely on aggregated measures of human annotations. Hence, sentiment analysis tools specifically trained on human-annotated data do not necessarily match human perception of aggressiveness, and corresponding results need to be taken with a grain of salt. By reporting our results and experience, we aim at confirming and raising additional awareness of this methodological challenge when studying aggressiveness (and sentiment, in general) in the software-engineering domain.

1. Introduction

Software development is a social activity involving substantial collaboration and communication among developers. Understanding communication in and across software developer teams is perceived a key element for research on how to attain project health (Cataldo and Herbsleb, 2013; Ehrlich and Cataldo, 2012; Grinter et al., 1999; Herbsleb et al., 2006; Kraut and Streeter, 1995). Developer communication comes in various forms and takes place over different channels (Storey et al., 2017). A popular communication channel in Open-Source Software (OSS) projects are mailing lists (Mauerer et al., 2022; Ramsauer et al., 2019). The conversational tone used in e-mails has an important influence of feeling welcomed as a newcomer and can be reason to stop contributing to a project (Ferreira et al., 2021; Steinmacher et al., 2013), as communication issues could harm new developers retaining in the project (Canfora et al., 2012; Steinmacher et al., 2019). Miscommunication as well as negative attitude and communication tone could intimidate developers in OSS projects (Ferreira et al., 2021; Storey et al., 2017). Recent research showed that issue discussions about respectfulness in OSS projects led to an increased developer turnover and to a decreased number of newcomers in these projects (Jamieson et al., 2024).

The huge and popular OSS community that develops the Linux kernel with its main communication channel, the *Linux Kernel Mailing List*[1] (LKML), is known for aggressive e-mails of some contributors when discussing technical issues (Alami et al., 2019; Egelman et al., 2020; Ferreira et al., 2021; Schneider et al., 2016). However, such aggressive e-mails could be perceived differently. For example, let us have a look at the following e-mail excerpt:

> **E-mail from the LKML:** *[...] WHY THE HELL DID YOU SEND THIS CRAP TO ME? [...]* [2] (Schneider et al., 2016)

While some may perceive this question as aggressive, others may not. That is, the perception of aggressiveness is human judgment (see Section 2, for more background information on aggressiveness and related concepts). As this small example already demonstrates, perceptions of aggressiveness can be highly individual and subjective. Consequently, understanding how aggressiveness is perceived is a key element for research on how to attain project health.

[1] https://lkml.org/ (accessed: 2024-04-08)
[2] https://lkml.iu.edu/hypermail/linux/kernel/1305.0/01484.html (accessed: 2024-04-08)

When we started working on this topic, our overarching goal was (and still is) to study aggressiveness in developer communication, to obtain insights into the consequences of aggressive communication, and to find measures to mitigate them. However, by means of a human annotation study on the Linux Kernel Mailing List, we found that the agreement of humans on aggressiveness is generally low. Therefore, instead of analyzing the consequences of aggressive communication on OSS projects, this paper is about the methodological challenge that we faced when analyzing the perception of aggressiveness in developer communication. Thus, this is an unusual paper. But, first things first.

1.1. Communication on the Linux Kernel Mailing List

Linus Torvalds, the founder and lead maintainer of the LINUX kernel, "is known for using strong language and sometimes insulting comments"[3] (Schneider et al., 2016). In 2018, after lots of private and public discussions about codes of conduct and toxic community behavior, he admitted that his behavior may have hurt other contributors and "possibly drove away from kernel development entirely"[4]. As a consequence, Linus Torvalds took a break on maintaining the LINUX kernel ("I am going to take time off and get some assistance on how to understand people's emotions and respond appropriately"[4]). This turning point in the LINUX kernel community motivated us to investigate the effect of aggressive language on the participation in and organizational structure of OSS projects, in general. In retrospective, we wanted to understand what the consequences of aggressive behavior are for project success and developer collaboration.

As a foundation of these and similar questions, it is necessary to automatically detect aggressive language on the LKML (and other communication channels), to which nearly 300 messages are sent per day (Schneider et al., 2016). As there is lots of related work on sentiment analysis tools, some of them even specifically designed and improved for the software-engineering domain (see Section 3), we wanted to find the most appropriate of these tools for our use case. For this purpose, we aimed at evaluating well-established state-of-the-art sentiment analysis tools against a ground truth in form of a sample of human annotated e-mails from the LKML, to ensure that the tools' results are valid. This is where we started our endeavor on assessing developer aggressiveness in e-mail communication, and, sadly, this is also where it ended. In this paper, we tell the story why this was the case. In particular, we want to share the methodological challenge that we faced, to allow other researchers to avoid similar problems.

[3] https://www.linux-magazine.com/Online/News/Linus-Torvalds-Takes-a-Break-Apologizes/ (accessed: 2024-04-08)

[4] E-mail from Linus Torvalds to the LKML on 2018-09-16: https://lkml.org/lkml/2018/9/16/167/ (accessed: 2024-04-08)

1.2. Overview of the Methodological Challenge

At the beginning of our study, we sampled 720 e-mails from the LKML for human annotation, as a foundation of our and other studies. Each of the sampled e-mails was annotated by multiple (6 to 9) human annotators of our team and beyond, all of them being involved in software engineering. Following empirical standards, we computed the inter-rater agreement across all annotated e-mails. However, we noticed that the inter-rater agreement was very low. Even when computing the agreement separately for different groups of annotators (experience, gender) and excluding few specific outliers, the agreement remained low. This was the turning point at which we started to investigate more deeply for which e-mails different annotators have different perceptions of whether an e-mail is aggressive or not, trying to identify patterns that could explain the generally low agreement. After extensive qualitative investigations and discussions with the human annotators, covering different aspects of our annotation study, we accept and understand that different human individuals perceive aggressive language differently, ending up in ambiguous annotations, without a common theme among them.

The picture emerges that sentiment analysis in the software-engineering domain is very difficult, as both, humans and tools, are not fully reliable to derive an appropriate ground truth on sentiment and aggressiveness. Due to the low agreement among humans, we cannot reliably evaluate the tools. Thus, this paper is not about analyzing the effect of aggressive language on developer collaboration and organizational structure in OSS projects, but about *the methodological challenge* to find a common ground for the perception of aggressiveness in communication among developers. So, based on the annotation results of 720 e-mails from the LKML, we address the following research objective:

> *Why is the inter-rater agreement among humans so low?*

Or, more generally speaking:

> *What are possible causes for ambiguous perception of aggressiveness and sentiment in the software-engineering domain?*

1.3. Ambiguous Perception of Sentiment

There is lots of considerable work in the field on sentiment and emotion analysis in the software-engineering domain (Novielli and Serebrenik, 2019), covering the development and improvement of sentiment analysis tools (see Section 3.1.1) as well as

applying such tools to answer questions regarding human and social factors and their effect on software projects (see Section 3.1.2). Such work relies on a common understanding and perception of aggressiveness (and other emotional communication aspects such as toxicity, polarity, etc.)—but such a common ground is difficult to establish, as we show in our study. When annotating texts by multiple people, it is publicly assumed that there are few deviants having different perceptions than the majority, which are often downplayed by using majority voting or measures of central tendency (e.g., Blaz and Becker (2016); Calefato et al. (2017); Gachechiladze et al. (2017); Guzman et al. (2017); Kaur et al. (2018); Kenyon-Dean et al. (2018); Novielli et al. (2018a); Panichella et al. (2015); Serva et al. (2015)).

When looking at the inter-rater agreement of the human annotators of our study, we were surprised that perceptions of human annotators are that diverse. However, looking outside of the software-engineering domain, this does not come as a surprise, because ignoring deviants is problematic and drawing implications from collected sentiment data is limited (Kenyon-Dean et al., 2018; Liu, 2010; Van Atteveldt et al., 2021). In other words, different treatment of deviants might considerably affect results. With making our endeavor public, we want to draw attention to the ambiguous perception of written human language with a focus on the software-engineering domain, showing that relying on a common perception of sentiment and deriving implications from it needs to be taken with a grain of salt. There is a need for a community effort for investigating human annotations on sentiment and human perception at large scale. With a community effort, new standards, guidelines, and metrics shall be developed in task forces or research seminars to obtain more reliable results from human annotation studies in the software-engineering domain; just using a few human annotators and combining the rating into a single score using state-of-the-art methods is not enough to derive general conclusions. This is also why we did not end up in answering the questions regarding the effect of aggressive language on the participation of OSS projects that were the initial motivation for our study.

1.4. Contributions

In summary, we make the following contributions:

- An overview of work on sentiment analysis in the software-engineering domain (including tool development, tool applications, and human annotation studies in software-engineering research) based on a systematic literature review;
- A set of 720 annotated e-mails from the *Linux Kernel Mailing List* for identifying aggressive language in the LINUX kernel community, where each of these e-mails was annotated by multiple (6 to 9) human annotators;
- A confirmation of the work of Imtiaz et al. (2018) and Herrmann et al. (2022), which both showed that human annotators have a low inter-rater agreement on sentiment in software-engineering texts;
- Insights into why the inter-rater agreement among humans is generally low, based on qualitative investigations in various directions, and a discussion of potential implications of this result for the research community.

Overall, our contributions provide a more nuanced view on human perception and sentiment analysis in the software-engineering domain. This way, our work expands the epistemical body of knowledge on sentiment analysis in the software-engineering domain, in that it confirms previous research results on a different dataset and with a different methodology. This constitutes a crucial part in scientific research to avoid spurious results, and, with our work, we call attention to the methodological challenge in this field, which should become an important part of future research.

Data Availability: We provide our annotation data (i.e., the set of e-mails used for annotation and the corresponding results), the scripts used for data preprocessing and evaluation, as well as detailed information about our systematic literature review on a supplementary website: `https://se-sic.github.io/paper-lkml-aggressiveness/` and `https://zenodo.org/records/14621553` .

2. Background

The analysis of communication tone and sentiment is a research area on its own, mostly outside of the software-engineering community. According to Liu, *sentiment* "is the underlying feeling, attitude, evaluation or emotion associated with an opinion" (Liu, 2017). Often, these terms are used interchangeably, although there are slight differences (Liu, 2020). Sentiment usually has a *target*, for example, a person, organization, topic, etc. "the sentiment has been expressed upon" (Liu, 2017). However, *emotions* do not necessarily have a target. Moreover, emotions are usually very short episodes in the brain that exist only in the moment in which the emotion is evoked, whereas sentiment usually lasts for a longer period of time (Munezero et al., 2014). *Sentiment analysis*, also called *opinion mining*, is the process of analyzing and extracting people's sentiment on written text (Liu, 2020).

The *polarity* of a sentiment describes whether the sentiment is positive, neutral, or negative, which is subjective and context dependent. In addition, the sentiment polarity can be of different intensity. For example, two texts can be both negative, but one could be

more negative than the other (Liu, 2020). This is usually determined by different ratings on a discrete scale, for example, from -2 to $+2$, where -2 means strongly negative, -1 weakly negative, 0 neutral, $+1$ weakly positive, and $+2$ strongly positive (Liu, 2017).

In computational linguistics, the term *toxicity* stands for negatively perceived verbal behavior, such as offensive language, bullying, harassment, or discrimination; but also kinds of mild aggression, stereotyping, or sarcasm fall into this category (Bhat et al., 2021). However, research has shown that the perception of mild aggression, stereotyping, or sarcasm is mostly context dependent and, therefore, difficult to detect (Pavlopoulos et al., 2020; Waseem et al., 2017).

In social psychology, *aggression* is defined as "behavior that is intended to harm another individual who does not wish to be harmed" (Baron and Richardson, 1994; Stangor et al., 2014). Rancer and Avtgis (2006) define *aggressiveness* as an "attack [to] the self-concepts of individuals instead of, or in addition to, their positions on issues". Aggressive behavior can either be of physical form (i.e., "use of the body to apply force" (Rancer and Avtgis, 2006)) or of verbal form (i.e., "use of words" (Rancer and Avtgis, 2006)). Infante and Wigley (1986) describe *verbal aggressiveness* as an "exchange of messages between two people where at least one person [...] attacks the self-concept of the other person in order to hurt the person psychologically." Verbal aggressiveness "can be even more harmful and long lasting than the results of physical aggression" (Buss, 1971; Infante and Wigley, 1986). When we refer to aggressiveness in this paper, we always mean verbal aggressiveness. Harm can be intentional or unintentional, where intentional harm is usually perceived worse. However, both, intentional as well as unintentional harm, and, thus, aggressiveness are associated with negative sentiment (Ames and Fiske, 2013; Stangor et al., 2014). Hence, sentiment analysis techniques are commonly used to identify aggressive language in written text, and detecting aggressiveness is considered a sub-task of sentiment analysis (Del Bosque and Garza, 2014).

In psychology, a differentiation is made between instrumental and reactive aggressiveness (Berkowitz, 1993; Scarpa and Raine, 1997). Whereas the former is "relatively nonemotional" and "directed toward obtaining some goal" (Scarpa and Raine, 1997), the latter is also called emotional aggressiveness, which often emerges as "defensive reaction in response to some perceived frustration, insult, or provocation" (Scarpa and Raine, 1997). Thus, emotional aggressiveness often appears together with anger (Scarpa and Raine, 1997). Aggressive language is also called "offensive, vulgar, opinionated, and rude" (Hamilton, 2012). Sometimes, it is also described as "unfriendly" or even "malicious" (Burroughs and James, 2005). However, impolite behavior, in general, also includes *incivility*, that is, nonverbal communication (Hamilton, 2012).

Associated with aggressive language and toxicity is *hate speech*, which is a verbal attack that has a specific target without any explanation and often incites violence (Fortuna and Nunes, 2019). It is often based on stereotypes and usually targets a group of people (e.g., minorities) instead of individuals (Fortuna and Nunes, 2019). Thus, hate speech is not specifically in the focus of our study. Instead, we aim at investigating any kind of aggressive language, independent of its target. Accordingly, we do not differentiate hate speech from aggressive language.

To detect aggressive language in texts, usually human annotators label the texts manually. Holm (1980) found that, when humans label aggressiveness in texts, their labels also depend on whether they can identify reasons or intents for the aggressive behavior. If they noticed that an aggressive behavior occurred as a response as kind of a self-defense, the level of aggressiveness was rated lower than when they assumed that somebody had another intent for being aggressive. But, as with toxicity, the perception of aggressive language is subjective. Also the definitions of the labeling categories as well as the annotation guidelines can introduce bias into individuals' perceptions (Garg et al., 2023). Consequently, different annotators may come to different labels (Pang and Lee, 2005). While the annotators often have a high agreement on whether a text contains an emotion or not, there is disagreement on the kind and intensity of the recognized emotion (Mohammad et al., 2015). To mitigate this problem, two different views are employed: On the one hand, there is *group perception* (e.g., the majority vote of the different annotators) and, on the other hand, there is *individual perception* of the annotator, which may deviate from the group perception (Kocoń et al., 2021). Hence, to get a complete picture, it is not enough to consider just the group perception, as a person's individual characteristics (e.g., mood, cultural or demographic background, sense of humor, etc.) can influence the individual perception (Kocoń et al., 2021). Still, standard procedures in sentiment analysis consist of removing controversial annotation data (Van Atteveldt et al., 2021). Often, just the majority vote is considered when humans do not agree. This is problematic, as Kenyon-Dean et al. (2018) showed on Twitter sentiment data: More than 30% of their samples are controversial among human raters. Such controversial samples should neither be ignored nor be assigned to a category based on the majority vote, but should be moved to a separate "complicated" category (Kenyon-Dean et al., 2018). Eventually, language and sentiment (in particular, aggressiveness) as well as their perception in society are subject to change over time, and, thus, there is still a "gap in social and computational understanding of toxicity" (Garg et al., 2023).

Beside human annotation, there are also various automated text-analysis approaches (D'Andrea et al., 2015) that are used as classifiers for automatically detecting sentiment polarity in texts. Recent research has shown, though, that human annotation performs better than automatic approaches based on machine learning (Van Atteveldt et al., 2021). Nevertheless, even human-annotated data need to be validated because of the potential disagreement of humans (Van Atteveldt et al., 2021). Hence, both, annotation-based approaches and automated text-analysis approaches are problematic. In general, sentiment analysis tasks are challenging, and state-of-the-art solutions are very limited due to different subjective perceptions of a text (Liu, 2010).

3. Sentiment Analysis for Software Engineering: A Literature Review

Sentiment analysis on software-engineering-related texts is even more complicated, as they contain technical content (e.g., natural language interleaved with code snippets, etc. (Mäntylä et al., 2018)) and vocabulary that is used in a non-standard way ("to kill" is negatively connotated in standard language, but is neutral in the software-engineering domain when talking about terminating processes) (Jongeling et al., 2017; Novielli et al., 2015). To obtain an overview of related work and to show how prominent sentiment analysis for the software-engineering domain is, we conducted an extensive literature review. The goal of our literature review is twofold: (1) By means of providing an overview of the state of the art, we demonstrate that the detection and analysis of sentiment in developer communication is a highly relevant but also challenging research topic in the software-engineering domain. Even more, we want to particularly stress the different challenges and inconsistent results that have been reported in the literature on sentiment analysis in the software-engineering domain, independent of whether human annotation was involved or not. (2) We aim at finding studies that have used human annotation for sentiment analysis in the software-engineering domain. This way, we obtain an overview of the different characteristics of the various human annotation studies on software-engineering texts and of the particular annotation methods that have been used.

We base this literature review on two pillars: First, we used three recent and established literature studies (Lin et al., 2022; Obaidi and Klünder, 2021; Obaidi et al., 2022b) on the development and application of sentiment analysis tools in the software-engineering domain as a first starting point and selected relevant papers. All three literature studies found that sentiment analysis tools for the software-engineering domain are frequently applied, different datasets are used to train the tools, and there are various challenges and limitations when training and applying the tools (e.g., specific terms are used differently in the software-engineering domain than outside this domain, irony cannot be properly detected by tools, or tools vary in their performance when using various datasets or evaluation strategies). In contrast to these literature studies, we focus on challenges that are relevant to and encountered in human annotation studies. However, since the established literature studies are already rich sources of information on which we can apply our own exclusion criteria, we use them as first pillar for our own literature review. Second, we complemented the set of the first pillar with our own systematic literature search on this topic. This way, we complement the existing literature studies to obtain a broader view with more details on how sentiment analysis is used in software-engineering research. For this purpose, we performed a GoogleScholar search as well as a search in the proceedings of highly-ranked software-engineering conferences (i.e., ICSE, ESEC/FSE, ASE)[5] using the search terms *"sentiment analysis in software engineering"* and *"toxicity in software engineering"*, and collected relevant papers via manual inspection of the paper content. In addition, we recursively collected relevant papers that have been referenced in relevant papers (also known as backward snowballing (Jalali and Wohlin, 2012)), again via manual inspection of the paper contents of the referenced papers. This way, we cover a variety of different venues (conferences, journals, workshops).[6]

From the papers that we collected from either pillar, we only included peer-reviewed papers written in English that analyze the sentiment (or similar concepts) of written developer communication in the software-engineering domain. That is, after our manual inspection of the paper contents, we applied the following **exclusion criteria**:

- Papers that are not written in English.
- Papers that have not been peer reviewed.
- Papers that deal with sentiment analysis in general, but are not related to software engineering.
- Papers that investigate user reviews, which only rate apps but do not cover discussions with developers.

[5] The definitions of conferences' and journals' abbreviations etc. are available on our supplementary website.

[6] We did not perform forward snowballing because we already found a large and diverse set of relevant and recent papers without. Although we had to exclude 44% of the papers that we found in our literature review according to our exclusion criteria, 34% of the remaining papers had not been found by the three established literature studies. More importantly, 20% of all the found papers that fulfill our inclusion criteria have been published in the years 2022 or 2023. As our goal was to obtain an overview of the methodological challenges identified in human annotation studies rather than obtaining a complete set of related papers, we have decided not to carry out forward snowballing on top.

Table 1
Overview of the used literature studies and of our own systematic literature review.

	# Papers found	# Papers marked as relevant (i.e., that fulfilled our inclusion criteria)
Obaidi and Klünder (2021)	80	63
Obaidi et al. (2022b)	107	85
Lin et al. (2022)	183	81
Distinct union of Obaidi and Klünder (2021), Obaidi et al. (2022b), and Lin et al. (2022)	234	117
Additional papers found by us	81	60
Overall sum	315	177

- Papers that perform content extraction or requirements classifications on software-engineering-related texts, as these tasks do not analyze the communication of developers.
- Papers that analyze developer communication but are not related to sentiment or similar concepts.
- Papers that analyze non-written (e.g., oral) developer communication.

We provide descriptive statistics of the used literature studies and our own systematic literature review in Table 1. The literature study of Obaidi and Klünder (2021) covers 80 papers, 7 of which deal with sentiment analysis, in general, but that are not related to software engineering. Another 10 papers of their study investigate user reviews, which only rate apps but do not cover discussions with developers. Consequently, we consider these papers not relevant for our study. As a result, we collected 63 papers from the literature study of Obaidi and Klünder. In an extended version (Obaidi et al., 2022b) of their literature study, they added 27 more papers. From these 27 papers, we excluded 5 papers that analyzed user reviews and included the remaining 22 papers. Independent of the two previously mentioned literature studies, Lin et al. (2022) conducted a literature review on opinion mining in the software-engineering domain, focusing on the analysis of opinions rather than the analysis of sentiment. From their 183 papers, we extracted 81 papers that fulfilled our inclusion criteria. After removing duplicates from all the three literature studies, we ended up with 117 papers that met our inclusion criteria out of 234 papers. In addition, in our own subsequent literature search, we found 81 papers that are related to sentiment analysis in the software-engineering domain and that were not found by the three previous literature studies on this topic. Out of these 81 papers, we included 60 papers that fulfilled our inclusion criteria. In total, when combining the papers from the previous literature reviews and from our own systematic literature search, we viewed 315 papers, out of which 177 fulfilled our inclusion criteria. On our supplementary website, we provide additional tables that contain entries for all the 315 viewed papers (separately for the 177 included and 138 excluded papers).[7] There, we also provide a reason for each excluded paper why we excluded it.

In sum, after combining papers from both pillars, our collection of papers contains 177 papers that have been published between 2013 and 2023 in highly-ranked software-engineering conferences and journals (ICSE, MSR, EMSE, etc.)[5], as well as in a variety of specialized workshops, conferences, and journals (e.g., SEmotion), see Table 6 in the Appendix.

Similar to Obaidi and Klünder, we grouped the relevant papers into two major categories (see Table 5 in the Appendix): Papers that *develop or evaluate* sentiment analysis tools for software-engineering tasks (Appendix A.1), and papers that merely *apply* sentiment analysis tools on software-engineering-related texts to answer research questions related to sentiment (Appendix A.2). Notably, in the study of Obaidi and Klünder, only one third of the papers address tool development and evaluation for the software-engineering domain. In our combined collection of papers, we observe a similar division between these two categories, but with a slightly higher fraction (39%) of papers that address tool development and evaluation than in their study. Also note that we treat human sentiment labeling (i.e., manual annotation) as a sentiment analysis tool, which is why we categorized 8 papers that simply used human labeling to answer research questions related to sentiment as tool application (see also Table 5 in the Appendix).

Whereas Obaidi and Klünder focus on quantitative results (e.g., how often which tool was used), we concentrate on rather qualitative results: We gather details on which tools have been developed specifically for the software-engineering domain, provide concrete examples how sentiment analysis is used in the software-engineering domain, and discuss that different studies led to contrary results for answering the same research questions. Although tool development and application is not directly related to human annotation, such studies still are a rich source of information for characterizing potential problems and challenges in sentiment analysis in the software-engineering domain. Some of these challenges that have been encountered in studies on tool evaluation are similar to the challenges that we encountered in our human annotation study, which is why we decided to also include such studies in our literature study. Moreover, many tool-evaluation papers rely on human-annotated datasets as a ground truth for their evaluation. Therefore, in our literature study, we discuss potential problems by means of

[7] https://se-sic.github.io/paper-lkml-aggressiveness/literature_table.html

concrete examples and provide detailed insights into how human data labeling was performed by different studies, which is not part of the literature study of Obaidi and Klünder.

In what follows, we first summarize the general findings that are related to our human annotation study, which we have extracted from our literature review. Then, we provide detailed information on the human annotation studies that we identified through our literature review.

3.1. General Insights and Related Work from the Literature Review

During our literature review, we obtained a number of general insights for sentiment analysis in the software-engineering domain. Although these insights are primarily gathered from tool-development or tool-application papers and not necessarily related to human annotation studies, the described challenges and results are still important and related to our study. Therefore, we first summarize the general findings that we have extracted from the tool-development and tool-evaluation papers, and then we summarize the general findings that we have extracted from the tool-application papers.

3.1.1. Insights from Tool Development and Evaluation

Jongeling et al. (2015, 2017) investigated whether sentiment analysis tools from outside the software-engineering domain agree with each other when used on technical texts, resulting in different sentiment classifications for different tools. They also compared the classifications' outcomes against a human-annotated dataset from Murgia et al. (2014), resulting in a disagreement between tools and humans for up to 60% of the texts with human agreement.

Novielli et al. (2015) annotated a StackOverflow dataset regarding emotions and opinions. They found that sentiment polarity is a complex phenomenon, which varies depending on recipients and technical matters. In later studies, they came to the conclusion that "reliable sentiment analysis in software engineering is possible" when existing tools are specifically tuned to the software-engineering domain (Novielli et al., 2018a,b).

Hence, many researchers developed their own sentiment analysis tools specifically tuned to the software-engineering domain, based on and evaluated on manually labeled datasets such as code review comments, issue comments, ticket systems, or StackOverflow posts. In Appendix A.1, we provide a brief overview of the different tools (we collected more than 30) and their related papers that we found through our literature review, and we provide further details about all these papers and tools on our supplementary website.

The fact that so many tools have been developed specifically for the software-engineering domain demonstrates, on the one hand, that analyzing developers' sentiments is a highly relevant topic. Yet, on the other hand, the huge number of different approaches, datasets, and tools also indicates that existing approaches may not be accurate and reliable enough.

Beside creating new tools, researchers compared the software-engineering-specific sentiment analysis tools against each other.[8] Multiple tools have been compared with different sets of human-annotated data (originating from multiple annotators; most of these sets ignore texts with disagreement after discussion among the annotators or use measures of central tendency (e.g., Blaz and Becker, 2016; Gachechiladze et al., 2017; Guzman et al., 2017; Kaur et al., 2018; Mansoor et al., 2021; Novielli et al., 2020, 2018a, 2021; Serva et al., 2015; Uddin and Khomh, 2021)) from the software-engineering domain, resulting in that these tools have a low overall accuracy and are not suited to detect negative sentiment in software-engineering texts (Biswas et al., 2019; Calefato et al., 2019; Cassee et al., 2022; Ferreira et al., 2021; Fucci et al., 2021; Shen et al., 2019). This even holds for more advanced machine learning and language models (Batra et al., 2021; Biswas et al., 2020; Bleyl and Buxton, 2022; Cabrera-Diego et al., 2020; Ferreira et al., 2024; Imran et al., 2022; Kadhar and Kumar, 2022; von der Mosel et al., 2023; Prenner and Robbes, 2022; Robbes and Janes, 2019; Wu et al., 2021). Interestingly, Sun et al. (2022) noted that the way how humans perceive and label sentiment in software-engineering-related texts (which are used for tool training and tool evaluation) "plays an important role for the performance of automated sentiment analysis".

Novielli et al. (2021) observed that different software-engineering-specific sentiment analysis tools led to contradictory results, which often result from different treatments of context, domain-specific words, politeness, indicators of positive and negative sentiments next to each other, or figurative language (Biswas et al., 2020; Chen et al., 2021; Novielli et al., 2018b; Uddin et al., 2022b; Uddin and Khomh, 2021). Even combining different sentiment analysis tools that have been trained on different kinds of software-engineering-related texts does not necessarily lead to more reliable results (Mula et al., 2022; Obaidi et al., 2022a). Also, adding or removing context (e.g., quotes of previous messages) can affect the performance of the sentiment analysis tools, but does not lead to a general improvement in their accuracy (Ferreira et al., 2024). In addition, many tools ignore emojis, which are used by authors of a comment to explicitly express their sentiment in a "self-reported" way (Chen et al.,

[8] We provide further information about all these papers as well as the corresponding tools and approaches on our supplementary website.

2021; Park and Sharif, 2021). Lin et al. (2018) even "warn[ed] the research community about the strong limitations" of such tools, which is corroborated by Wang (2019). Others, in turn, proposed guidelines how these tools could be used "reliably" (Novielli et al., 2020, 2018b) in the software-engineering domain, and that there is a "substantial agreement" among the tools that are specifically developed for this domain (Ahasanuzzaman et al., 2020).

In contrast to the work that considers sentiment analysis tools as reliable and closest to our paper, after manually annotating 589 GitHub comments, Imtiaz et al. (2018) came to the conclusion that sentiment analysis in the software-engineering domain is unreliable, as "human raters also have a low agreement among themselves". They evaluated 6 sentiment analysis tools and observed that neither the tools agreed among each other nor did they agree with the consensus (which was achieved after discussing the disagreeing annotations) of their human raters. Their results are in line with our study. In fact, we are able to confirm their general results on a different dataset, which is a valuable contribution on its own. In addition to that, the main difference to our study (except for data source and data sampling) is that, in their study, only two human raters annotated each comment, whereas we put our analysis on a broader basis by having 6 to 9 human annotators per text.

Similarly to our study, Herrmann et al. (2022) analyzed the human perception of 100 statements from pre-labeled and widely-used datasets from GitHub, JIRA, or StackOverflow. To that aim, they asked 94 participants to label each of these statements, resulting in a huge difference between the labels assigned by the different participants. Only in 7 statements they achieved a substantial agreement between the pre-defined labels from the datasets and the majority vote of their participants. Noteworthy, none of their 94 participants agreed with all of the 100 predefined labels from the datasets. In our study, we use a different software-engineering-related dataset (e-mails from the LKML), a larger number of texts (720 e-mails), but a lower number of annotators (6 to 9) per text. Despite these deviations between their study and our annotation study, we can confirm their results on the subjectivity of human perception, that is, the lack of agreement between human annotators. Even more, in a qualitative study, we investigate why the disagreement on specific texts is particularly high.

In summary, even though researchers constantly developed new sentiment analysis tools for the software-engineering domain in the last decade, the accuracy of these tools is rather low. Notably, many of the tool evaluations rely on specifically annotated datasets, which different humans may perceive differently, though. All in all, this shows that sentiment analysis in the software-engineering domain is a long-burning issue and, still, a hot topic in software-engineering research.

3.1.2. *Insights from Tool Application and Tool Usage*

As discussed in Section 3.1.1, there are lots of sentiment analysis tools specifically designed for software engineering. Beside their development and evaluation, these tools have also been used in various studies to empirically answer specific research questions. In Appendix A.2, we provide a brief overview of all these studies, which we have collected during our literature review. More details are available on our supplementary website. In what follows, we report on a selection of these studies that we consider most closely related to the results of our annotation study:

In general, emotions and sentiment polarity are present in the communication channels of OSS projects (Ferreira et al., 2019c; Graßl and Fraser, 2022; Guzman and Bruegge, 2013; Jurado and Rodriguez, 2015; Murgia et al., 2014; Tourani et al., 2014), but usually only a small fraction of the communication expresses a positive or negative sentiment (Ferreira et al., 2019c; Hata et al., 2022; Sengupta and Haythornthwaite, 2020; Skriptsova et al., 2019; Valdez et al., 2020). Whereas most GitHub projects are neutral, there are 10% more projects with negative sentiment than projects with positive sentiment (Sinha et al., 2016).

In addition, specific events (e.g., receiving feedback from another developer or disagreement on how to implement a specific feature) can cause a change in a developer's sentiment (Garcia et al., 2013; Li et al., 2021). The role of such events is critical for a project, as up to 24% of the developers who received negative feedback never contributed to the project again (Freira et al., 2018). In particular, the sentiment that is prevalent in the replies that newcomers (i.e., new developers joining a project) receive influences whether they continue contributing to the project (Mahbub et al., 2021). Nevertheless, developers seem to be less influenced by negative sentiment than users, and replies often continue the emotion of the initial message (Lanovaz and Adams, 2019). However, many developers also try to resolve conflicts and reply in a neutral or polite manner after receiving a comment that contains negative sentiment (Ortu et al., 2016a). Even comments with negative sentiment are not only criticizing, but often also constructive (Assavakamhaenghan et al., 2023). Moreover, positive or negative sentiment in organizational discussions seems to be related to changes in the socio-technical structure of a software project and, thus, also has an impact on the sustainability of the project (Yin et al., 2023).

As another factor, the discussion topic seems to impact developers' sentiment. Specifically, discussing

non-reproducible bugs or security-related issues is associated with negative sentiment (e.g., due to annoyance or frustration) (Goyal and Sardana, 2017; Pletea et al., 2014; Rahman et al., 2015).

Researchers also investigated the sentiment of commit messages and build processes, having partly contrary results (Huq et al., 2019, 2020; Islam and Zibran, 2016, 2018c; Kaur et al., 2022; Souza and Silva, 2017; Venigalla and Chimalakonda, 2021b). For example, whereas two studies found that commit messages and issue comments written on Mondays are more negative than on other days (Guzman et al., 2014; Kumar et al., 2022), other researchers found that the most negative sentiment is present in comments written on Tuesdays (Sinha et al., 2016), and still another study reports that comments written on Tuesdays are least negative, but those that are written on Sundays are most negative (Valdez et al., 2020). Also the time of day can affect the sentiment of a comment: Whereas positive sentiment seems to occur most frequently in the morning and least frequently shortly before midnight, negative sentiment is prevalent throughout the whole day (Valdez et al., 2020).

However, whether developers perceive code reviews as toxic seems to be subjective (Chouchen et al., 2021). Nevertheless, when multiple developers post comments to the same issues, their comments often share the same sentiment (Li et al., 2020).

Further studies on GitHub issues investigated various characteristics of toxic comments (Cheruvelil and da Silva, 2019; Cohen, 2021; Ferreira et al., 2022; Miller et al., 2022; Raman et al., 2020), indicating that only a small fraction is aggressive (Cohen, 2021; Ferreira et al., 2022).

Closest to our aim, Ferreira et al. (2019a) assessed on the LKML whether the maintainers' sentiment changed after Linus Torvalds's temporary break. However, they did not find any significant changes between 2017 and 2019. Later on, Ferreira et al. (2021) collected 1 545 e-mails from the LKML that were related to rejected patches. In their study, two persons manually coded the e-mails with respect to incivility, ending up in a substantial agreement between the two coders. They identified potential causes for incivility in these e-mails and evaluated how much incivility does exist.[9] In our study, we sampled 720 e-mails from the LKML and had 6 to 9 humans who annotated each e-mail with respect to aggressiveness. In contrast to Ferreira et al. (2021), we observe a disagreement between our human annotators on a substantial number of e-mails. In addition, we identify potential causes for the different perceptions of aggressiveness among humans (i.e., among the annotators) and discuss why individual perceptions matter, whereas Ferreira et al. (2021) focused on understanding the communication between the developers (who authored the collected e-mails) and analyzed causes for incivility in general (not causes for different perceptions of aggressiveness).

It is important to note that many researchers already pointed out that the existing sentiment analysis tools face further challenges. For example, Kritikos et al. (2020) stated that irony detection does not work properly. Murgia et al. (2018) concluded that specific keywords, such as "thanks" or "sorry", are important for sentiment analysis tools to detect sentiment polarity correctly. Further, Ferreira et al. (2019b) mentioned another challenge: Sometimes, a single sentence contains contradicting sentiments, which makes it difficult to determine an overall sentiment for the whole sentence.

3.2. Human Annotation Studies

Throughout our literature review on sentiment analysis for software engineering, we found a substantial number of papers in which a human annotation study on software-engineering-related texts was conducted (either for tool evaluation prior to tool selection, or to qualitatively investigate the sentiment in such texts). In particular, 55 out of the 177 papers that fulfilled our inclusion criteria contain manual data labeling.[10] Note that manual data labeling is a cross-cutting methodological aspect that occurs in papers from both categories, "tool development and tool evaluation" as well as "tool application and tool usage".

To perform the manual data labeling, different text-preprocessing strategies, a different number of human raters, and different aggregation methods to obtain a single label from multiple ratings have been used. In Table 2, we provide an overview of the different approaches that have been used by the papers that we found in our literature study. The vast majority of papers uses two annotators for each text snippet, followed by three annotators. Murgia et al. (2018) even claimed that "using more than two raters does not significantly change the results in terms of degree of agreement on emotions". Nevertheless, there are also several studies that used 4 to 16 annotators for each text snippet. Surprisingly, there are also a couple of studies that rely on the perception of only a single annotator (Biswas et al., 2020; Herrmann and Klünder, 2021; Klünder et al., 2020; Qiu et al., 2022). According to Qiu et al. (2022), Egelman et al. (2020) have shown that "inter-rater agreement is very high when using multiple annotators, implying that a single annotator is sufficient" (Qiu et al., 2022).

[9] Although Ferreira et al. (2021) investigate potential causes for incivility and we investigate potential causes for aggressiveness on the LKML, we consider their study on incivility related to our study, since incivility is a kind of unfriendly behavior that considers nonverbal communication (Hamilton, 2012), but their study basically considers verbal communication as we do in our study on aggressiveness.

[10] We marked the particular 55 papers that contain manual data labeling in our literature table on our supplementary website: https://se-sic.github.io/paper-lkml-aggressiveness/literature_table.html

Table 2
Number of annotators per text snippet, annotation labels, and methods for disagreement resolution used in the human annotation studies of the papers that we included in our systematic literature review.

Characteristics of the Human Annotation Studies		Papers
Number of annotators per text snippet	94	Herrmann et al. (2022)
	3–16	Murgia et al. (2018); Ortu et al. (2016b)
	10	Mansoor et al. (2021)
	6	Park and Sharif (2021)
	5	Patwardhan (2017)
	4	Cohen (2021); El Asri et al. (2019); Gachechiladze et al. (2017); Kaur et al. (2018)
	3–4	Uddin and Khomh (2021)
	2–4	Murgia et al. (2014)
	3	Ahmed et al. (2017); Blaz and Becker (2016); Calefato et al. (2018, 2017); Chouchen et al. (2021); Guzman et al. (2017); Islam and Zibran (2018b); Novielli et al. (2018a); Rong et al. (2022); Wang (2019)
	2–3	Batoun et al. (2023); Ding et al. (2018); Hata et al. (2022); Lin et al. (2019); Novielli et al. (2020); Serva et al. (2015)
	2	Cassee et al. (2022); Cheriyan et al. (2021); Ferreira et al. (2021); Fucci et al. (2021); Graßl and Fraser (2022); Imran et al. (2022); Imtiaz et al. (2018); Lin et al. (2018); Mäntylä et al. (2017); Marshall et al. (2016); Novielli et al. (2021); Sanei et al. (2021); Sapkota et al. (2019); Sarker et al. (2023a, 2020); Sengupta and Haythornthwaite (2020); Tourani et al. (2014); Uddin et al. (2022a)
	1–2	Ferreira et al. (2022); Robe et al. (2022)
	1	Biswas et al. (2020); Herrmann and Klünder (2021); Klünder et al. (2020); Qiu et al. (2022)
	multiple (unspecified)	Miller et al. (2022)
Annotation labels	positive, negative, neutral	Ahmed et al. (2017); Batoun et al. (2023); Biswas et al. (2020); Blaz and Becker (2016); Calefato et al. (2018); Ding et al. (2018); El Asri et al. (2019); Graßl and Fraser (2022); Herrmann and Klünder (2021); Herrmann et al. (2022); Klünder et al. (2020); Lin et al. (2018); Mansoor et al. (2021); Novielli et al. (2020, 2018a); Park and Sharif (2021); Patwardhan (2017); Sanei et al. (2021); Uddin et al. (2022a); Uddin and Khomh (2021)
	positive, negative	Lin et al. (2019); Tourani et al. (2014)
	positive, negative, neutral, mixed	Novielli et al. (2021)
	positive, negative, neutral, mixed, sarcasm	Imtiaz et al. (2018)
	negative, non-negative, mixed	Cassee et al. (2022); Fucci et al. (2021)
	toxic, not toxic	Chouchen et al. (2021); Cohen (2021); Miller et al. (2022); Qiu et al. (2022); Raman et al. (2020); Sarker et al. (2020)
	five-point Likert scale (very pos. to very neg.)	Guzman et al. (2017); Sapkota et al. (2019)
	scale from 1 to 9	Mäntylä et al. (2017)
	self, other, object	Gachechiladze et al. (2017)
	personal, racial, swearing	Cheriyan et al. (2021)
	love, joy, surprise, anger, sadness, fear	Calefato et al. (2017); Imran et al. (2022); Kaur et al. (2018); Murgia et al. (2014); Ortu et al. (2016b)
	multiple different categories	Ferreira et al. (2021); Hata et al. (2022); Robe et al. (2022); Rong et al. (2022); Sengupta and Haythornthwaite (2020); Serva et al. (2015); Wang (2019)
Disagreement resolution	majority vote	Calefato et al. (2017); Gachechiladze et al. (2017); Guzman et al. (2017); Kaur et al. (2018); Mansoor et al. (2021); Murgia et al. (2014); Novielli et al. (2018a); Park and Sharif (2021); Patwardhan (2017); Serva et al. (2015); Uddin and Khomh (2021); Wang (2019)
	majority vote, ignore if tie	Islam and Zibran (2018b)
	consensus discussion	Batoun et al. (2023); Cassee et al. (2022); Cohen (2021); Ding et al. (2018); El Asri et al. (2019); Ferreira et al. (2022); Fucci et al. (2021); Hata et al. (2022); Imran et al. (2022); Imtiaz et al. (2018); Rong et al. (2022); Sanei et al. (2021); Sarker et al. (2023a, 2020); Sengupta and Haythornthwaite (2020); Tourani et al. (2014); Uddin et al. (2022a)
	consensus discussion, ignore if unresolved	Novielli et al. (2020, 2021)
	majority vote + consensus discussion	Blaz and Becker (2016); Calefato et al. (2018)
	manual resolution by uninvolved people	Lin et al. (2019, 2018)
	median of numeric labels	Herrmann et al. (2022)

That is, they assumed that it is enough to have one annotator and human resources can be saved when a high agreement between the labelers can be assumed. However, in our study, we show that such an approach is not necessarily reliable and particularly risky, as we collect a substantial number of e-mails to which multiple annotators assigned contrary labels.

Prior to their annotation studies, the different researchers applied different text-preprocessing strategies. In many studies, URLs (Biswas et al., 2020; Calefato et al., 2018; Imran et al., 2022; Sanei et al., 2021), HTML tags (Biswas et al., 2020; Calefato et al., 2018; Sanei et al., 2021), code snippets (Biswas et al., 2020; Calefato et al., 2018; Imran et al., 2022; Imtiaz et al., 2018; Kaur et al., 2018; Klünder et al., 2020; Sanei et al., 2021; Tourani et al., 2014), and stack traces (Kaur et al., 2018; Tourani et al., 2014) are removed from the software-engineering-related texts beforehand. Imtiaz et al. (2018) also removed emojis, and Tourani et al. (2014) also filtered auto-generated e-mails. Moreover, Klünder et al. (2020) and Imran et al. (2022) also removed user mentions, names, or e-mail addresses and replaced them by anonymous tokens. Nevertheless, Sanei et al. (2021) showed that removing emojis, numbers, method names, punctuation, or replacing uppercase letters by lowercase letters even led to more disagreement than when keeping these elements as they are in the original texts.

Regarding the treatment of context, there exist different approaches. While some studies removed the context and citations within text snippets (Lin et al., 2018; Sanei et al., 2021; Tourani et al., 2014), others kept the context, such that it is available during annotation (Ferreira et al., 2021; Imtiaz et al., 2018; Serva et al., 2015), since providing context helps to avoid mislabeling (Uddin and Khomh, 2021). Murgia et al. (2018, 2014) compared annotations in which the context or citations were available with annotations on the same text snippets where the context and citations were removed. They concluded that "context can cause doubt" (Murgia et al., 2014) for the annotators and slightly decreases the inter-rater agreement compared to annotations in which the context was removed beforehand (Murgia et al., 2018), but, in general, it "does not play a significant role" (Murgia et al., 2014).

As the majority of comments or similar texts in the software-engineering domain are neutral, Ahmed et al. (2017) applied an undersampling of neutral comments, whereas Sarker et al. (2020) and El Asri et al. (2019) explicitly oversampled toxic or negative comments, respectively.

Also, there are different ways on how to train the annotators prior to the actual annotations. Few studies consciously forwent any training to capture the annotator's real, unbiased perception of sentiment (Ahmed et al., 2017; Ding et al., 2018; Mansoor et al., 2021; Murgia et al., 2018). Hence, they neither had discussions with the annotators prior to annotation, nor did they provide any annotation guidelines. In contrast, a substantial corpus of studies provided specifically developed annotation guidelines prior to the annotation study (Biswas et al., 2020; Calefato et al., 2018; Cassee et al., 2022; El Asri et al., 2019; Fucci et al., 2021; Hata et al., 2022; Herrmann et al., 2022; Imtiaz et al., 2018; Novielli et al., 2018a; Sarker et al., 2020). Some studies even had training sessions in which they discussed the guidelines with the annotators (Novielli et al., 2018a; Uddin and Khomh, 2021), or provided tutorials with supervised annotation beforehand (Blaz and Becker, 2016; Calefato et al., 2018). Herrmann et al. (2022) showed that providing guidelines leads to more reliable results than ad-hoc labeling, as the guidelines enable a common understanding of the labels among all annotators.

Table 2 shows that most of the studies used three labels for annotation (positive, negative, neutral), whereas some explicitly omitted the neutral label to enforce a binary decision (Lin et al., 2019; Tourani et al., 2014), and others added a mixed label that allows positive and negative sentiment to be simultaneously present in one text snippet (Imtiaz et al., 2018; Novielli et al., 2021). Also having multiple labels that capture different emotions or having special labels to answer special research questions (e.g., the target of an attack (Gachechiladze et al., 2017)) are quite common. However, annotations that consist of multiple labels of emotions are sometimes mapped back to polarity labels (positive, negative, neutral) (Kaur et al., 2018; Uddin et al., 2022a). In addition to the actual labeling, Serva et al. (2015) also asked each annotator for the level of confidence on a four-point Likert scale for each label they assigned.

When multiple annotators annotate the same text snippet, they can perceive them differently and, thus, may assign different labels. Most of the studies used discussions among the annotators to achieve a consensus in case of disagreeing labels after individual annotation. Also majority votes are often used. Some studies even combined discussion and majority vote (Blaz and Becker, 2016; Calefato et al., 2018), and others explicitly ignored text snippets where the annotators, even after discussion, cannot agree on a label (Novielli et al., 2020, 2021), or the majority vote ends up in a tie (Islam and Zibran, 2018b). In some cases, also people that were not involved in the annotation study were asked to either label the texts with disagreement individually or to take a decision based on the disagreeing annotations (Lin et al., 2019, 2018). Uddin and Khomh (2021) mentioned that they asked an additional annotator to solve the disagreement "in cases of sarcasm and convoluted polarity". In addition, it is important to note that the different studies report different levels of detail on their annotation

Table 3
Overview of our design choices and the corresponding papers from the literature.

Design Choice	Papers Found in Our Literature Review
Oversampling of aggressive comments	El Asri et al. (2019); Sarker et al. (2020)
E-mail preprocessing: removing URLs, etc.	Biswas et al. (2020); Calefato et al. (2018); Imran et al. (2022); Sanei et al. (2021)
Replacing names and e-mail addresses by tokens	Klünder et al. (2020)
Removing auto-generated e-mails	Tourani et al. (2014)
Removing citations	Ferreira et al. (2019a); Garcia et al. (2013); Lin et al. (2018); Rousinopoulos et al. (2014); Sanei et al. (2021); Tourani et al. (2014)
Selection of annotation labels	see the different options in Table 2
Annotation guidelines	Biswas et al. (2020); Calefato et al. (2018); Cassee et al. (2022); El Asri et al. (2019); Fucci et al. (2021); Hata et al. (2022); Herrmann et al. (2022); Imtiaz et al. (2018); Novielli et al. (2018a); Sarker et al. (2020)
Tutorials with supervised annotation	Blaz and Becker (2016); Calefato et al. (2018)
Number of annotators (6–9)	e.g., Mansoor et al. (2021); Murgia et al. (2018); Ortu et al. (2016b); Park and Sharif (2021)
Disagreement resolution	see the different options in Table 2

studies, and they all achieve different levels of agreement among their annotators, reaching from substantial disagreement to substantial agreement. Only few studies investigated the disagreements and searched for possible reasons why the annotators had chosen different labels.

In summary, the literature indicates that there are many different design choices when conducting human annotation studies in the software-engineering domain, including the usage of different annotation labels, numbers of annotators per text snippet, preprocessing techniques, methods to prepare the annotators, and methods for disagreement resolution. When making the design choices for our annotation study, which we present in the following section, we oriented towards the choices that have been used in related studies, ensuring that our study builds on proven design choices of related studies. In Table 3, we provide an overview of our design choices and list the corresponding papers from our literature review that took a similar choice.

4. Methodology

In our empirical study, we have sampled 720 e-mails from the LKML and annotated whether they are aggressive or not. In what follows, we describe how we have collected the e-mail data, how the annotation study was conducted, and how we have processed and analyzed the annotation results.

4.1. Data Collection

We downloaded the e-mails of the LKML from the publicly available mailing-list archive GMANE[11] using the tool NNTP2MBOX[12], covering 1 939 567 e-mails in the time period from June 2005 to July 2016. From all these e-mails, we extracted the content and automatically removed parts that are not plain text, such as attachments, diffs, commit messages (containing a tag `Signed-off-by`), keys, etc. Afterwards, we removed e-mails that contain less than 5 alphabetic characters (to get rid of e-mails that just contain GIT version numbers) and e-mails that contain only auto-generated content (indicated by a field `Git-Commit-Id:` or `Robot-Id:` in the e-mail header).

4.2. Sampling of 720 E-Mails

As we aimed at finding a ground truth to assess the performance of sentiment analysis tools on aggressive and non-aggressive e-mails, we sampled 720 e-mails from the collected 1 939 567 e-mails for human annotation. This is a compromise to obtain a substantial and diverse dataset, and, at the same time, not to overburden the annotators (each of which was expected to annotate every e-mail of our dataset). As the majority of e-mails of the LKML, usually, may not contain aggressive language (only about 7% of the e-mails sampled by Ferreira et al. (2021) were uncivil), we oversampled e-mails of people who are known to be likely aggressive or are involved in publicly known flame wars and other sensational conversations, based on manual web search. That is, we sampled e-mails that have been sent by or to persons that were publicly mentioned to be participating in a flame war (or similar events) in the months prior, during, and after the flame war became public. Due to protection of privacy, we cannot publicly state from which people we have oversampled the e-mails.[13] In sum, we collected 652 e-mails from, at least, 15 distinct selected people, also including e-mails for which we fixed the recipient (i.e., the `To:` header field) to people who have publicly reported to be a victim of offensive language, sometimes even restricted to a certain time period of two to three years in which certain flame wars, etc. had occurred. In addition to these

[11] https://gmane.io/, list: `gmane.linux.kernel` (accessed: 2019-02-12)

[12] https://github.com/xai/nntp2mbox/ (accessed: 2019-02-12)

[13] Additional information can be provided upon personal request, though.

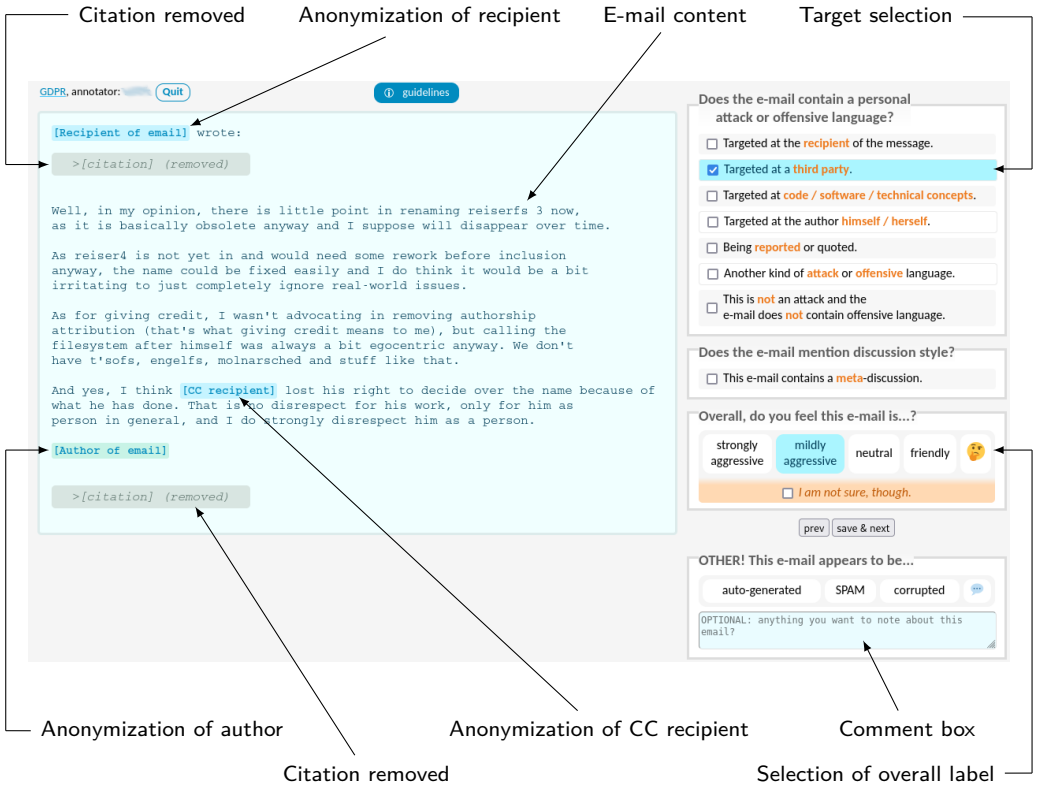

Figure 1: Example screenshot of our annotation tool (with additional explanations surrounding the screenshot).

652 deliberately (but still randomly with respect to the above described constraints) sampled e-mails, we also sampled 68 e-mails completely random. Note that, on purpose, we sampled only e-mails that were replies to previous e-mails on the LKML (considering the `References:` or `In-Reply-To:` header fields), as we assumed that aggressive behavior merely occurs when a community member reacts or replies to a previous e-mail. Altogether, we sampled 720 e-mails authored by 111 different developers. Then, we randomized the order of the e-mails prior to our annotation study, but we showed all sampled e-mails to all annotators in the same order.

4.3. E-Mail Preprocessing

To avoid annotators being biased, we anonymized the names of authors and recipients, e-mail addresses, and URLs that appeared in the e-mail content. We replaced them by the special tokens `[Author of e-mail]`, `[Recipient of e-mail]`, `[CC recipient]`, `[e-mail address]`, and `[URL]`, similarly to the work of Klünder et al. (2020). This way, it is not obvious to the annotators who had authored a particular e-mail or who else had participated in the conversation. In addition, we removed citations within e-mails, that is, content which originally came from another e-mail. Citations are identified by a > character at the beginning of a line. We removed citations since we wanted to assess the aggressiveness of the e-mail (i.e., the text the author wrote), and not aggressive parts that have been written by another person. Removing citations in e-mails has also been used in previous work (Ferreira et al., 2019a; Garcia et al., 2013; Lin et al., 2018; Rousinopoulos et al., 2014; Sanei et al., 2021; Schneider et al., 2016; Tourani et al., 2014). Analyzing the language of cited parts may be interesting to understand why somebody reacted in an aggressive way, but this is not necessary to identify aggressive language of the e-mail author. Hence, we replaced citations by a token `>[citation] (removed)` to indicate for the annotators at which places we have removed cited content.

4.4. Annotation Tool

To conduct our annotation study, we created a small web application to manually annotate e-mails. In a small pre-study, two of the authors of this paper and two former colleagues evaluated different interfaces of this tool to find suitable options for annotations. For each e-mail, we presented the preprocessed e-mail content on the left part of the screen (scrollable in case of longer e-mails) and the annotation options on the right (see a screenshot of an example e-mail in Figure 1). In the upper part, annotators are asked to select the target of a personal attack or offensive language. Inspired by the study of Wulczyn et al. (2017), we provided the options “targeted at the recipient”, “targeted at a third party”, “targeted at code

/ software / technical concepts", "targeted at self", "being reported or quoted", "other kind of attack or offensive language", and "none". In addition, annotators could mark an e-mail as "meta" if they identify an e-mail discussing communication issues. Finally, the annotators had to choose an overall label: "strongly aggressive", or "mildly aggressive", or "neutral", or "friendly". We provided the possibility to choose between different degrees of aggressiveness to account for individual perceptions, but we provided only four labels to enforce the annotators to either choose an aggressive label or a non-aggressive label. If the annotators believed that an e-mail is auto-generated (e.g., a message that was created by an automated response service, though we have removed most of the auto-generated e-mails beforehand), somehow corrupted, or spam, they could just choose these separate options without deciding upon the aggressiveness of an e-mail.

During our pre-study, we noticed that some annotators were not sure which label to choose. In such a case, they might think about their decision for a comparably long time. To also account for this case, we decided to provide an additional check box "I am not sure": If an annotator has already chosen a target but is still struggling with the choice of the overall label for a certain amount of time (dependent on the number of e-mail lines to read) or if an annotator alters the already chosen overall label, the "I am not sure" checkbox appears automatically. Independent of that, annotators always had the possibility to add a comment. After finishing the annotation of one e-mail, using the next button brought the annotator directly to the next e-mail. However, at any point in time, it was possible to navigate back to previous e-mails and to change the annotation of a previously seen e-mail (to allow corrections if an annotator identifies later that an earlier assigned e-mail should be labeled differently than initially labeled).

4.5. Annotation Guidelines

On the one hand, Ross et al. (2016) showed that it is important to provide clear definitions for the possible labels to ensure a high inter-rater agreement. Consequently, several human annotation studies in the software-engineering domain developed annotation guidelines (Biswas et al., 2020; Calefato et al., 2018; Cassee et al., 2022; El Asri et al., 2019; Fucci et al., 2021; Hata et al., 2022; Herrmann et al., 2022; Imtiaz et al., 2018; Novielli et al., 2018a; Sarker et al., 2020). On the other hand, other human annotation studies in the software-engineering domain forwent providing annotation guidelines to capture the annotators' real and unaffected perception (Ahmed et al., 2017; Ding et al., 2018; Mansoor et al., 2021; Murgia et al., 2018).

For our annotation study, we provided clear explanations and examples for each annotation label to all annotators in our annotation guidelines and in a tutorial (see also Section 4.6). After finishing our pre-study, which we had conducted to evaluate and improve the interfaces of our annotation tool, the authors and the remaining participants of the pre-study developed the annotation guidelines and annotation examples together during multiple discussions. Instead of providing scientific definitions that would have been rather difficult to understand and hard to hold them in the annotators' heads, we focused on providing light-weight, exemplary definitions. In addition, we provided multiple examples for each label (called *annotated samples*), on which all pre-study participants agreed, and provided detailed explanations on why a specific label applies to a specific example.[14] In particular, our annotated samples contain extreme cases as examples to explain the different targets, overall labels, and other possible choices. This way, we aimed at conveying our impressions of what the differences between the different labels are and how the labels should be chosen during annotation to the annotators, to obtain a common understanding of aggressiveness as far as practicable, without restricting the annotators in their own perceptions.

4.6. Annotation Study

We performed our annotation study on 720 emails sampled from the LKML, split in two batches of 360 e-mails each. Splitting the dataset was a design decision to obtain the possibility to react to unforeseen problems that could potentially occur during the study. To that aim, there was a break of several months between the two batches, which we used to check the data formats. Fortunately, we only detected minor tooling issues after the first batch (e.g., stored data formats, etc.) that did not affect our results. Some annotators only participated in the first batch, others only in the second batch; most annotators participated in both batches. In total, 10 different annotators participated in our annotation study.

As previous studies have shown that especially newcomers in a software project are influenced by negative sentiment (Canfora et al., 2012; Ferreira et al., 2021; Jamieson et al., 2024; Steinmacher et al., 2019, 2013), we decided to let outsiders judge the aggressiveness of the 720 e-mails, instead of asking developers of the Linux kernel (who also could have been biased by annotating e-mails they have authored themselves, they have read beforehand, or they have additional knowledge of due to their development activity in the Linux kernel community). Thus, each e-mail was manually annotated by 6 to 9[15] annotators

[14] We provide the annotation guidelines and the set of annotated samples (in the way in which we presented them to the annotators) on our supplementary website.

[15] Some of the annotators were not able to label all e-mails due to personal time restrictions. Thus, we have a different number of annotations for individual e-mails.

from our group and beyond (e.g., students, Ph.D. students, external participants), all of them having programming experience, but at different levels. At the beginning, we queried some background information from all annotators, covering gender, programming experience on a scale from 1 (novice) to 10 (expert), estimated programming experience compared to colleagues on a scale from 1 (junior) to 5 (senior), and the number of OSS projects they already have contributed to. 4 of our annotators were female, 6 male. 3 stated that they are in the age of 18–24, 6 were 25–34 years old at the time of annotation, one did not reveal their age. The chosen experience values range from 3 to 9 with a median at 6. When asked about their experience compared to others, values between 2 and 4 were chosen (median 3). We asked for this background information to be able to investigate whether people's gender or experience makes a difference in their perception of aggressive language.

Before starting to annotate the e-mails, the annotators were asked to take a look at our annotation guidelines (including our annotated samples) and they had to take a mandatory tutorial, similar to previous annotation studies in the literature (e.g., Blaz and Becker, 2016; Calefato et al., 2018). In the tutorial, we showed 10 e-mails as iconic examples of the possible choices (e.g, of aggressive language). During the tutorial, the annotators had to annotate an e-mail first, then we showed them which labels the authors of this paper would have chosen (with a short textual explanation), to provide some exemplary feedback on what the annotators' task is and on when to choose which target, etc.[16] After an annotator finished the mandatory tutorial, the actual e-mail annotation began (as explained in Section 4.4). The annotators were asked to carefully read each e-mail and then (1) decide whether this e-mail contains personal attacks against a specific target and (2) choose the overall label on whether the e-mail is strongly aggressive, mildly aggressive, neutral, or friendly (as explained in Section 4.4). Note that the annotation guidelines as well as the annotated samples were available to the annotators also during the entire annotation study.

The annotators had to annotate the e-mails independently of each other. As our annotation tool was developed as a web application, it was their own choice when to log into the tool and annotate the e-mails. We asked them to perform the annotations within a couple of weeks, knowing that annotating all e-mails takes several hours per batch. The annotators were allowed to pause annotating at any time; at their next login, the annotation tool automatically continued at the e-mail at which they had stopped before. Moreover, the annotators were allowed to work at their own pace. So, they could take as much time as they needed to read an e-mail, think about it, and annotate it. We made this design decision deliberately, not to put the annotators under time pressure during annotation. Since the annotation time is not informative, we also did not track the time spent on annotation.

[16] On our supplementary website, we provide all the e-mails from the tutorial together with the exemplary feedback that the annotators received after their annotation in the tutorial.

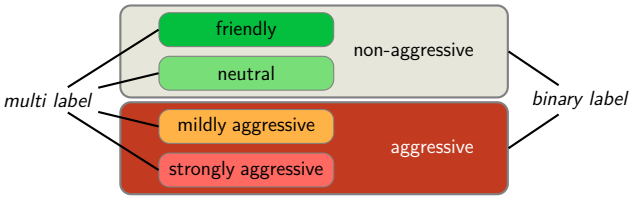

Figure 2: Overview of the multi-label approach that we have used for human annotation, and our mapping to binary labels that we applied afterwards to each annotation.

As we noticed after the end of our study that, for some of the e-mails, the annotated labels varied considerably among the different annotators, we interviewed them why they had decided that certain e-mails are aggressive or neutral/friendly while other annotators came to the opposite decision. During this discussion, some of the annotators decided to adjust their label for some of these e-mails when they did not agree any more with their previously chosen label, which might have been chosen as a result of fatigue. For the majority of these e-mails, however, the annotators maintained their decision, even when the opinions were in stark contrast to others. In addition to the individual interviews, we also arranged a consolidation meeting, in which we discussed the controversial e-mails together. During this consolidation meeting, all annotators agreed that others might perceive an e-mail differently than themselves, but stayed with their previously chosen label. We qualitatively explore the controversial e-mails later on in Section 6.

4.7. Label Mapping & Inter-Rater Agreement

As the overall labels are rather fine-grained and as there may be only a slight difference between some of the labels, we derived a binary label for each annotation: Similarly to the work of Ahmed et al. (2017), we combined *neutral* and *friendly* to *non-aggressive* (0), and we combined *mildly* and *strongly aggressive* to *aggressive* (1), as social studies indicated that a lower number of classes leads to a higher inter-rater agreement (Salminen et al., 2018). We provide a graphical overview of our label mapping from the multi label to the binary label in Figure 2.

In a next step, we computed the inter-rater agreement, for both, multi label and binary label. To this end, we use Krippendorff's alpha for ordinal data (Krippendorff, 2019), which is an established and widely-used inter-rater reliability measure (Díaz et al., 2023) that quantifies the agreement among a

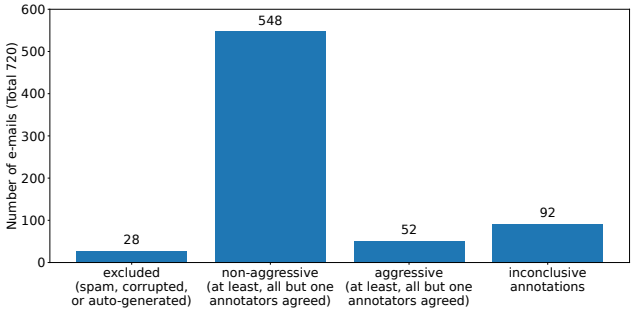

Figure 3: General overview of the results of our annotation study on 720 e-mails from the LKML (excluding unsure annotations, that is, annotations that have been marked as "I am not sure" by the annotator).

set of raters. Krippendorff's alpha is able to deal with missing data and different numbers of annotations per e-mail (which is important for us as some of our annotators did not finish annotating all 720 e-mails, as stated in Section 4.6), and it is "sensitive to small sample sizes" (Krippendorff, 2019). Krippendorff's alpha is 0 when there is a "perfect disagreement" and 1 when there is a "perfect agreement" among the raters (Krippendorff, 2009).[17] There is an "acceptable level of agreement" for this measure: It is "customary to require" Krippendorff's alpha to be ≥ 0.8; for "tentative conclusions" a value ≥ 0.67 might be sufficient; below that, it is unreliable to draw conclusions from the data (Krippendorff, 2009).

We computed the inter-rater agreement for the complete dataset and, additionally, also for different subsets. In one group of subsets, we excluded all *unsure* annotations (i.e., annotations where the annotator marked the e-mail as "I am not sure"), see columns "excl. unsure" in Table 4. Beyond that, we computed the inter-rater agreement separately for different groups of annotators (i.e., male and female annotators, low and high experience, contributions or no contributions to OSS projects, etc.; see the first column of Table 4), to evaluate whether there is a higher agreement within a specific group of annotators even when the overall agreement is low.

5. Results of Our Annotation Study

In this section, we present the results of our human annotation study with regard to the overall labels chosen by the annotators, and we report the inter-rater agreement that we obtained from the human annotations.

5.1. Annotated Overall Labels

In sum, 720 e-mails have been annotated by 6 to 9 annotators each. From the 720 e-mails, we excluded 28 that have been marked as "spam", "corrupted", or "auto-generated" by, at least, 5 of the annotators (in most cases, all annotators agreed). That is, our dataset consists of 692 e-mails that are treated as real, human-written e-mails. If we take the mode of the binary labels of the annotations of an e-mail (considering a tie as non-aggressive, and excluding unsure annotations only if there are 4 or more sure annotations for an e-mail), 608 e-mails have been labeled as non-aggressive and 84 as aggressive. When excluding all unsure annotations, for 217 of the 692 e-mails, at least, one annotator labeled the e-mail as aggressive, and for the remaining 475 e-mails nobody labeled them as aggressive. If we consider only e-mails for which all but one annotators agreed on the same binary label (excluding all unsure annotations), 548 are labeled as non-aggressive, 52 as aggressive, and for 92 e-mails we are inconclusive about the overall aggressiveness as more than one person deviates from the annotation label of the majority (see also Figure 3). We analyze the content and annotations of these 92 e-mails later on in Section 6.

5.2. Inter-Rater Agreement

For the complete dataset of all annotations and e-mails, we computed Krippendorff's alpha as a measure of inter-rater agreement. Our results for both, the binary labels (aggressive vs. non-aggressive) and the multi labels (friendly, neutral, mildly aggressive, strongly aggressive), are listed in Table 4. Overall, as expected (see Section 4.7 and Salminen et al. (2018)), the inter-rater agreement is higher when using binary labels, as the different perceptions between friendly and neutral and between mildly aggressive and strongly aggressive are rather marginal. That is, there is a higher agreement on the general tendency (whether an e-mail is aggressive or not) than on the perceived degree of aggressiveness (e.g., whether it is mildly or strongly aggressive). For that reason, we focus only on the binary labels, as they seem to be more reliable. Nevertheless, Krippendorff's alpha is only around 0.5 (0.49 when considering all annotations, 0.52 when removing unsure annotations), meaning that the results are unreliable as the value is not even close to 0.67 (i.e., the threshold for tentative conclusions) (Krippendorff, 2009).[18] As a consequence, we grouped the annotators into several groups according to their gender, programming experience, and OSS contributions, to see whether there is a higher agreement in specific groups (see Section 4.7). In general, the agreement seems to be slightly lower for female

[17] For additional information on Krippendorff's alpha and its definition, see Krippendorff (2019).

[18] To avoid bias due to a single inter-rater agreement measure and a single threshold, we also used a second measure of inter-rater agreement, the Intraclass Correlation Coefficient (ICC). Also for the ICC, we receive similar results to Krippendorff's alpha: The inter-rater agreement is way beyond 0.75, which is the corresponding ICC threshold for good reliability (Koo and Li, 2016). We report the ICC next to Krippendorff's alpha on our supplementary website.

Table 4
Inter-rater agreement on the 720 annotated e-mails.

Group	Annotators per group ∗	Binary Label: all annotations K's α	Binary Label: excluding unsure K's α	Multi Label: all annotations K's α	Multi Label: excluding unsure K's α
All	10	0.49	0.52	0.43	0.44
Male	6	0.51	0.53	0.43	0.44
Female	4	0.37	0.43	0.45	0.49
Contribution to no OSS project	4	0.57	0.60	0.57	0.58
Contribution to ≥ 1 OSS projects	6	0.44	0.47	0.33	0.35
Contribution to ≤ 2 OSS projects	7	0.53	0.56	0.52	0.55
Contribution to > 2 OSS projects	3	0.46	0.48	0.35	0.34
Low experience (1–5)	4	0.46	0.51	0.48	0.52
High experience (6–10)	6	0.51	0.52	0.40	0.51
Compared to colleagues:					
Lower experience (1–2)	3	0.52	0.56	0.55	0.58
Higher experience (3–5)	7	0.46	0.49	0.38	0.39

∗ not all annotators have annotated all e-mails (we had 6–9 annotators per e-mail) K's α: Krippendorff's alpha

annotators (~ 0.4) than for male ones (~ 0.5), but for both groups the agreement is still low. Having a low or high programming experience does not make a big difference, either. When looking at the number of OSS projects the annotators had contributed to, we receive the highest agreement among the annotators that had no contribution to an OSS project before (0.57–0.60), but even this agreement is far below the thresholds for reliable agreement. To summarize, due to the generally low agreement, which we were surprised about, we later on started manual investigations on why the inter-rater agreement is so low (see Section 6).

6. Why Is the Inter-Rater Agreement among Humans so Low?

As we have found that the inter-rater agreement among our annotators was generally low, and as we have identified 92 e-mails (out of 720) where more than one person disagrees with the annotation result of the majority of annotators, we manually investigated the content and annotation results for these 92 e-mails. All the anonymized e-mail contents from our annotation study as well as all the results of our manual investigations are available on our supplementary website. When we provide excerpts of e-mails, in the following, the numeric identifier for the e-mail refers to the order in which the annotators labeled them; the complete content for these e-mails can be found on our supplementary website.

6.1. Investigation of Targets

As the annotators not only had to choose whether an e-mail is aggressive or not, but also had to choose an aggression target, we had a closer look at the annotated targets to gain deeper insights into why the annotators may not agree on an overall label. First of all, we computed the inter-rater agreement on the overall label for each e-mail individually and then grouped the e-mails by the mostly selected target of each e-mail. As the annotators had 7 different options, the chosen targets were very different, without any pattern or possible distinction appearing. Therefore, we grouped the targets into three categories (as there is usually higher agreement when there are fewer categories (Salminen et al., 2018)): *Human*, *Technical*, and *Other*, to investigate whether annotators agree or disagree more on aggression targeted at humans than at technical artifacts (such as source code), or vice versa. Hence, the targets "recipient", "third party", and "self" are treated as *Human* target, the target "code / software / technical concepts" as *Technical* target, and "being reported or quoted" and "other kind of attack or offensive language" are treated as *Other* target. Looking at our complete dataset of annotations, 164 e-mails have *Human* as mostly selected target, whereas 75 e-mails have *Technical* as mostly selected target. The remaining e-mails have either *Other* as the mostly selected target or no target at all. For the 92 e-mails with disagreement, we observe that 65 e-mails have *Human* as the mostly selected target, whereas 18 e-mails have *Technical* as mostly selected target. Grouping the e-mails into these target categories still results in generally low agreement on the overall label for each of the categories. For comparison, we computed these agreements separately for the 600 e-mails for which all but one agreed on the overall label and for the 92 e-mails with disagreement. For the former, the inter-rater agreement on the overall label of e-mails in target category *Human* (median 0.52) seems to be higher than for e-mails in target category *Technical* (median 0.16). However, for the

latter, these agreements are similar for e-mails in the target categories *Human* (median 0.16) and *Technical* (median 0.10). We neglected the category *Other* as only few (41 out of 720) e-mails received this category as mostly chosen target.

Since grouping e-mails into target categories, in general, did not reveal any significant differences in the inter-rater agreements for the e-mails with disagreement on the overall label, we investigated the targets and overall labels for the 92 e-mails of interest individually. We identified that only for 2 out of these 92 e-mails, a *Human* and a *Technical* target had been selected together; for the remaining e-mails, either only *Human* or only *Technical* targets had been chosen (neglecting *Other*). During our investigations, we noticed that there are e-mails for which the number of annotators who have selected targets *Human* or *Technical* is greater than the number of annotators who have selected the overall label "aggressive" for this e-mail. Such e-mails are interesting since some annotators acknowledged some form of aggression against a target despite saying the e-mail is not aggressive, which seems contradictory. In particular, we identified 76 e-mails of this sort (i.e., having an aggression target despite not being labeled as aggressive). So, in the next step, we looked at the overall label of these e-mails (see Figure 4). Particularly, we are interested in whether the aggressive annotations of these e-mails were mostly "mildly" or "strongly" aggressive. We found that most of the aggressive annotations of these e-mails were "mildly aggressive", which was expected as these e-mails are generally on the boundary between aggressive and non-aggressive. However, there are also e-mails with unusually many (i.e., more than one) "strongly aggressive" annotations. Consequently, we again looked into the contents of these 76 e-mails in detail, searching for common themes and to obtain a better understanding why different annotators have different opinions about these ambiguous e-mails, as we discuss next.

6.2. Manual Inspection

For our manual, qualitative inspection of the ambiguous e-mails, we use an open coding approach (Babchuk, 1996; Goulding, 2002) to search for possible causes for ambiguity. Our approach is similar to the open coding approach of Wang et al. (2023), who manually searched for possible causes for inconsistencies between emoji reactions and the detected sentiment in pull-request comments. In particular, the first two authors carefully read all the 76 ambiguous e-mails and looked for noticeable features in the e-mail content that might have led to the different opinions of the annotators and extracted these noticeable features as potential causes for ambiguity. Note that we did not find any noticeable features in some of the e-mails and, thus, no potential cause was extracted for

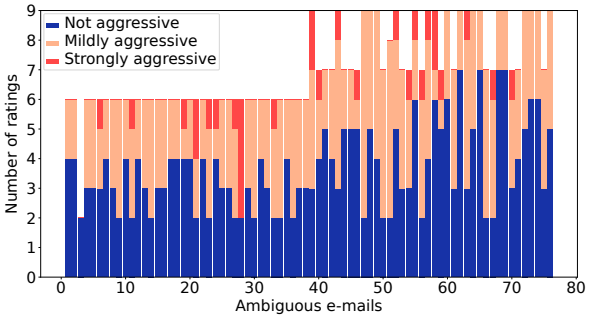

Figure 4: Overview of how many annotators had chosen which overall label (i.e., aggressive or non-aggressive), separately for each of the 76 ambiguous e-mails (i.e., the e-mails for which the number of annotators who had selected targets *Human* or *Technical* for an e-mail is greater than the number of annotators who had labeled this e-mail as aggressive).

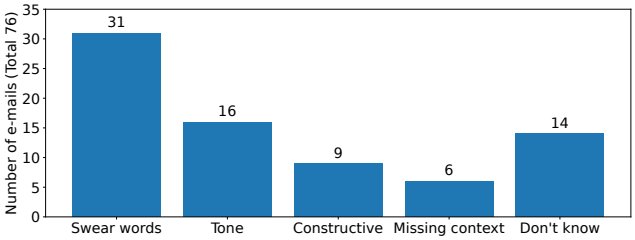

Figure 5: Potential causes for the ambiguity of the e-mails for which some annotators provided aggression targets but labeled them as "non-aggressive".

these e-mails. In the case that the first two authors were unsure about the presence of such noticeable features in an e-mail, this particular e-mail was discussed with the remaining authors to commonly derive a potential cause, if possible. After all the potential causes have been extracted from the 76 e-mails, we performed a second pass to assure consistency among the extracted potential causes of each e-mail. Via our manual, qualitative inspections, we identified four possible causes for ambiguity (see Figure 5), which, of course, depend on our interpretations and may be interpreted differently by others (see Section 7.1). We describe these causes in what follows:

Swear words. 31 e-mails contain swear words, which can be perceived rather ironic or funny, for instance, sometimes with regard to technical stuff, and therefore may have been annotated as non-aggressive by some people, whereas other people may perceive these swear words as aggressive.

> **E-mail 173:** *[...] it shouldn't be a fucking black magic, since fs folks really ought to understand what's going on there. [...]*

> **E-mail 126:** *[...] Read the fucking standard. [...] And yes, all quotes you've given are correct. [...]*

These e-mails, sometimes, also contain all-caps sentences, which may be perceived as screaming, which could also be the reason for aggressive perception.

Tone. 16 e-mails seem to have an attacking tone without using swear words. They might sound aggressive to some people, whereas the attacking tone still may sound neutral to other people as, for example, in the following excerpt:

> **E-mail 2:** *[...] it seems you're missing the whole point. [...]*

Constructive parts. For 9 e-mails, we identified an attack or aggression against somebody, but these e-mails also contain parts which could be perceived to have a constructive tone (e.g., the e-mail has lots of technical content and the author is willing to help solve a technical problem), and so the constructiveness of the e-mail may outbalance the aggressive part through the eyes of some annotators, whereas others may be clearly concerned about the attacking voice.

> **E-mail 537:** *What are you talking about? [...] You have no clue about how the b44 hardware works, do you? [...] Please explain in detail how ssb is wrong. [...]*

There are also e-mails that are completely constructive and use a friendly tone, where there is a P.S. at the end of the e-mail that contains some kind of attack or offensive language.

> **E-mail 550:** *[...] p.s. [Recipient of e-mail]. _don't_ do that again. i don't care who you are: internet archives are forever and your rudeness will be noted by google-users and other search-users - long after you are dead.*

Missing context. In 6 e-mails, we noticed that some information about the context is missing, for example, an e-mail referring to previous e-mails or depending on quotes (which we had removed, as explained in Section 4.3). So, if the e-mail is a reaction to a previous event, it may depend on this previous event whether the e-mail is considered aggressive or not. As sentiment polarity is context dependent (Liu, 2017) and as different annotators may imagine different contexts when the context is unknown, this could be a cause for ambiguous perceptions (although providing more context can also introduce more ambiguity (Murgia et al., 2018)). In E-mail 707, for example, some annotators see a strong exclusion of the recipient and find it aggressive because it suggests that the person is not part of the group, as one of our annotators pointed out in the interview after the annotation study, whereas others perceive the language as neutral. In fact, one cannot derive from the missing context whether this is appropriate language (e.g., a joke) or a serious exclusion.

> **E-mail 707:** *[...] the world you are living in is drastically different from the one where the rest of us lives.*

In some of the e-mails, a combination of missing context and abbreviations (probably unknown to the annotators) lets some of the annotators assume that an e-mail is aggressive, such as the "NAK" (negative acknowledgment) in the following e-mail:

> **E-mail 16:** *[...] NAK - it's too ugly to live.*

If the context is unknown, annotators are not sure how to treat the e-mail, leading to ambiguous labels. We also checked whether the usage of abbreviations could be a cause for ambiguous labels, but we did not find a common theme for abbreviations.

Unclear causes. For the remaining 14 (out of the 76) e-mails, we could not detect a potential cause for why these e-mails might be perceived ambiguously. One reason could be that there might be some form of aggressiveness that we (i.e., the authors of this paper) do not perceive, because human beings are very different when it comes to emotions and sentiment. Another reason might be that some annotators could have been absent-minded when annotating one of these e-mails and may have chosen an aggressive label inadvertently.

In summary, during our manual inspection, we identified possible reasons why some annotators have voted for "aggressive" and why others potentially not, but there is no common theme—there are individual reasons for each e-mail, which makes building a common ground truth difficult.

6.3. Other Factors that Could Have Influenced Human Perception

While we identified possible causes for ambiguous perception within the contents of the ambiguously labeled e-mails, other factors could also influence human perception, such as the environment and personal constitution of a person at the time of reading. For example, stress threatens developer participation in OSS projects (Raman et al., 2020), and, therefore, may lead to different perceptions of aggressiveness. However, as the annotators could take as much time for annotating e-mails as necessary and could pause in-between whenever they wanted, we consider potential stress not as a relevant factor in our study. Still, we checked whether the ambiguous e-mails have appeared at the end of our annotation study or whether there is a tendency for more aggressive or more non-aggressive votes over time, as this could be a notion of fatigue. However, we obtain a stable picture over time, for both batches of our annotation study: Chosen targets, overall labels, and the ambiguous e-mails themselves are distributed across the annotation order. In addition, we also checked whether splitting

up the dataset into two batches, having a break of several months in-between annotating the e-mails of the different batches, has affected the outcomes of our study results. Therefore, we investigated how many of the ambiguous e-mails were part of each batch: From the 76 ambiguous e-mails, 38 have been part of the first batch, and 38 have been part of the second batch. So, the percentage of ambiguously rated e-mails is equal in both batches.

Also, the length of e-mails may affect the annotation, since there are e-mails containing only one short sentence, while others contain lengthy continuous text. To account for this, we computed the word count of each e-mail and compared the annotation results of e-mails whose length is below the median with those whose length is above. Nevertheless, this did not reveal any noticeable differences in the annotation results.

7. Discussion & Perspectives

In this section, we first comment on the potential threats to validity of our study. Thereafter, we discuss our results in due consideration of the presented threats to validity. Finally, we provide perspectives on future work based on the insights that we obtained from our study.

7.1. Threats to Validity

As always in empirical studies, the validity of the results of our annotation study may be threatened in various ways. In the following, we discuss the potential threats to validity, grouped into different categories.

Internal Validity. Clearly, our results are dependent on our human annotators. Therefore, to reduce threats, we chose multiple annotators with different programming experience and gender. Although not all annotators finished annotating all e-mails of a batch (one of the annotators stopped after annotating 106 out of 360 e-mails, another one after 110 e-mails), we obtained 6–9 annotations per e-mail. We did not remove the annotations of the annotators who did not finish annotating, since not considering their perceptions would introduce an even more serious threat as we would thereby disregard specific perceptions. Note that, for the computation of the inter-rater agreement, the number of annotations per e-mail does not matter, as Krippendorff's alpha is capable of handling different numbers of labels per e-mail.

Also time and personal constitution (such as potential notions of fatigue that could appear toward the end of the annotation study after reading hundreds of e-mails) as well as e-mail length may affect the annotation, but our additional checks regarding annotation order and e-mail length did not reveal any noticeable differences, as discussed in Section 6.3.

Construct Validity. The annotation procedure could have been affected by the style and format of our annotation tool. We consider this a negligible threat, as we conducted multiple pre-studies to refine the annotation choices and presentation format. We also developed a mandatory annotation tutorial as well as annotation guidelines to form a common basis for the annotators. Even if the tutorial and annotation guidelines cannot rule out the possibility that one of the annotators slightly misunderstood the annotation task, we did not encounter any misunderstandings during the interviews and consolidation meeting with the annotators that we conducted after the end of the annotation study.

Removing citations in e-mails threatens the validity of our annotation study, as the missing context (which could be given by such citations) might have affected the annotators' choices. However, removing citations is common in the literature (Ferreira et al., 2019a; Garcia et al., 2013; Lin et al., 2018; Rousinopoulos et al., 2014; Sanei et al., 2021; Schneider et al., 2016; Tourani et al., 2014). In contrast, there are also studies in the literature that do not remove citations (Ferreira et al., 2021; Imtiaz et al., 2018; Serva et al., 2015; Uddin and Khomh, 2021). Consequently, there is disagreement in the literature on whether keeping citations is beneficial to avoid misclassifications, or whether removing context is beneficial to avoid that an annotator considers the sentiment of the citation instead of the sentiment of the actual text that should be annotated. Not only for human annotation, but also for training and evaluating sentiment analysis tools for the software-engineering domain, the role of context is largely unclear. Ferreira et al. (2024) investigated whether adding or removing context improves the accuracy of such tools. They found that adding context leads to a higher accuracy only for detecting some specific emotions, but not in general (Ferreira et al., 2024). Thus, we do not consider the removal of citations as an actual threat to the validity of our investigation on the low inter-rater agreement of human annotation, but rather as an uninvestigated area of the research topic that needs further investigation. In our human annotation study, we decided to remove citations, as previous work has shown that adding context has only a minor impact on human annotation results and causes slightly more *disagreement* among the annotators, since also the context could be perceived differently by different persons (Murgia et al., 2018, 2014).

Another threat may be caused by our e-mail sampling. Yet, even though the majority of e-mails (76%) was labeled as non-aggressive (as expected), our sample consisted of a substantial number of clearly aggressive or ambiguously labeled e-mails (20%), so our sampling strategy was suitable for our purposes. Without oversampling e-mails from people that are likely to be

aggressive, the percentage of aggressive e-mails would have been likely to be even lower, as previous studies have reported that only between 2% and 19% of their manually analyzed messages contained negative sentiment (El Asri et al., 2019; Ferreira et al., 2021; Uddin and Khomh, 2021). Therefore, it is also common in the literature to apply sampling strategies that try to oversample or undersample certain kinds of texts in order to increase the chance of sampling a significant number of texts with negative sentiment (Ahmed et al., 2017; El Asri et al., 2019; Sarker et al., 2020). Moreover, drawing conclusions about the perception of aggressiveness without sampling a substantial number of aggressive texts would be an even more serious threat to validity.

For the ambiguously labeled e-mails, we identified potential causes for the ambiguity in a qualitative analysis. Of course, this depends on the interpretations and perceptions of the authors of this paper and could be a potential threat, as other individuals may identify other causes. This, however, is in line with the central message of this paper: Human perception is different, and findings derived from it have to be taken with a grain of salt.

External Validity. Although the LINUX kernel is only one specific software project, our results may also apply to other projects, because other mailing lists and other communication channels behave similarly (Storey et al., 2017), and human disagreement is present in other contexts, as well (Imtiaz et al., 2018; Kenyon-Dean et al., 2018; Lin et al., 2018; Novielli et al., 2020). While other forms of communication, such as issue trackers or private chat messages, have different characteristics and different styles (e.g., they might be less formal than public e-mails and might not contain citations of previous messages, which is a common practice in e-mails), analyzing how humans perceive these messages faces similar methodological challenges. However, dedicated studies are necessary to investigate how different communication styles and channels affect the methodological challenges.

In our study, we particularly aimed at analyzing aggressiveness in developer discussions. Although we have, thereby, focused on a very specific kind of human behavior, the identified challenges are also generalizable to different kinds of emotions and sentiment, in general, since we have collected a substantial number of papers from the software-engineering domain in our literature review that either report contradictory results or related challenges when analyzing sentiments and emotions. Consequently, we deem our reported results on inter-rater agreement and methodological challenges not only important in our specific setting, but also consider investigating these challenges important for sentiment analysis in the software-engineering domain, in general.

7.2. Discussion of Our Results

Our results show that the perception of aggressiveness in technical texts from the software-engineering domain differs among individuals, even if they have a similar computer-science background. This poses a serious threat to validity for studies on sentiment and aggressiveness in software engineering. In fact, we were surprised by the variety of ratings for the e-mails in our comparably small dataset. Hence, we conclude that it is not sufficient to create a human-annotated dataset with a few annotators and to use measures of central tendency (e.g., majority voting) to get one label (which is widely used in the literature, see Section 3.2). Since the perception of aggressiveness is highly subjective, as the low inter-rater agreement in our annotation study shows, measures of central tendency disregard non-majority perceptions (e.g., even if 49% of the humans would perceive a text as aggressive, the majority vote only respects the perception of the 51% who perceive the text as non-aggressive). Instead, there is a need for a community effort for human annotations on sentiment perception, to obtain an overview of how people perceive aggression in our field and to develop methods that are capable of reliably handling and representing as many of the diverse perceptions as possible. That is, with community effort, large-scale analyses on the perception of sentiment and, in particular, aggressiveness in the software-engineering domain could be conducted. Using insights from large-scale analyses, new standards, guidelines, and data aggregation metrics for human annotation studies in the software-engineering domain could be developed with community effort. To achieve this, task forces and dedicated research seminars could be held to discuss the methodological challenge and work on suitable solutions for how to properly cover and handle diverse developer perceptions.

Diverse perceptions of aggression can become a problem when multiple persons collaborate, for example, on communication channels of OSS projects. If we want to understand how aggressive communication behavior influences participation and organizational structure of OSS projects, it is not sufficient to consider the opinion of the loud majority only. There could be also contributors with another perception who are intimidated by aggressive behavior, even if it is not perceived as aggressive by the author or most of the community. To form a healthy, welcoming atmosphere, individual perceptions need to be respected.

Notably, we identified multiple e-mails with (almost) a tie in the annotation labels regarding aggressiveness, even after discussion with the annotators. For example, irony can be perceived differently and can lead to inconsistencies in how people react (Wang et al., 2023); how the recipient perceives the message is highly individual. This has also been pointed out in previous research to be one of the pending challenges

for sentiment analysis in the software-engineering domain (Ferreira et al., 2024; Kritikos et al., 2020; Obaidi and Klünder, 2021). Ignoring such ambiguous texts upon dataset creation misses a substantial part of the community opinion. Combining human communication with technical collaboration even complicates communication processes and makes it more difficult, as some people seem to be less emotional with respect to technical artifacts than with respect to humans (Gachechiladze et al., 2017; Novielli et al., 2015), which is conflict-prone.

Our results not only affect the creation and usage of software-engineering-specific sentiment analysis tools, but also targets the field of trustworthy artificial intelligence. If sentiment analysis tools were specifically trained for a certain domain, such as software engineering, this does not necessarily mean that these tools match human perception about aggressiveness. Instead, we suggest to be more specific and identify specific perception triggers (i.e., locate certain text elements that could lead to ambiguous perception), which needs further research and maybe also requires expertise from outside the software-engineering domain.

Sentiment analysis for software engineering is not only an issue within research, but will also be relevant for practitioners when it comes to applying sentiment analysis tools in practice. For example, when such tools are used on social coding platforms (such as GitHub) to automatically detect inappropriate behavior and prevent aggressive authors from participating in discussions or code contributions, the tools need to be as accurate as possible. From what we have seen in our study, such tools should not rate overall aggression only, but also the likelihood that a message can be perceived as ambiguous. That is, instead of applying discrete labels, it could be necessary to develop continuous-scale metrics that consider both the perception of the majority of annotators and the subjective perception of individuals at the same time. However, to be able to develop such continuous-scale metrics, it is necessary to understand which parts of a text lead to aggressive perception, to find an appropriate weighting of group and individual perception, and to investigate how uninvestigated factors might potentially affect the perception of aggressiveness.

7.3. Perspectives on Future Work

Up until now, we have illustrated that human annotation studies in the context of aggressiveness in the software-engineering domain come with lots of troubles and difficulties. How does this help to overcome these issues? In the following, we present some avenues of future work on how we can improve on the state of the art based on our results.

Although we do not contribute directly actionable improvements with regard to sentiment analysis in this domain (which was also not the aim of our study), our aim is to call attention for the methodological challenge and ask the research community for help to find individual, appropriate solutions for the presented challenge. In particular, additional studies are necessary to move forward in several directions, especially with regard to causal investigations of human disagreement and to address the challenges that we have identified in our study, which we discuss below. This is not only a pure software-engineering-related problem: Similar challenges are also prevalent in other research areas that perform sentiment analysis (see Section 2). Garg et al. (2023) pointed out that there still is a "gap in social and computational understanding of toxicity". However, due to the special technical content that is part of software-engineering texts (such as text interleaved with code snippets or vocabulary usage in a non-standard way, as demonstrated in previous research (Jongeling et al., 2017; Mäntylä et al., 2018; Novielli et al., 2015)), we assume that bridging this gap is much more difficult and requires more detailed investigations in the software-engineering domain than in non-technical environments. Nevertheless, through our study, we already have obtained valuable insights and results that can be used to move forward in researching sentiment in the software-engineering domain.

The possible causes for ambiguous perception of aggressiveness that we have identified in our study can be used in future research for deeper investigations of developers' behavior, but also to improve annotation guidelines for human annotation studies, or for the development and improvement of sentiment analysis tools in the software-engineering domain. In what follows, we present possible avenues on how to address the identified challenges in further research:

- *Investigate how the annotation guidelines affect the annotations:* As we have shown in our literature study, human annotation studies used different ways on how to train the annotators (see Section 3.2). While previous work revealed that providing annotation guidelines leads to more reliable results than ad-hoc labeling (Herrmann et al., 2022), the influence of the extent and level of detail of such guidelines is still unclear. To explore the role of annotation guidelines, our annotation study shall be re-run with providing guidelines of different granularity and different comprehensiveness.
- *Investigate how context affects the annotations:* Since we have identified missing context as one of the potential reasons for low inter-rater agreement (see Section 6.2), there is a need for investigations on whether and how human annotations are affected by available or missing context. Although the role of context has been subject of multiple studies, their results are contradictory

or show significance only in very specific cases (see Sections 3.2 and 7.1). Thus, research should dedicate particular attention to the role context plays, especially in technical settings like on the LKML. To that aim, our annotation study shall be re-run while keeping context (i.e., citations of previous messages).

- *Investigate how certain text elements affect the annotations:* The results of our study indicate that an attacking tone and constructiveness, but also the use of swear words and abbreviations, could be potential reasons for disagreement (see Section 6.2). While studying the literature, we found that many different text preprocessings have been used for the evaluation of sentiment analysis tools (e.g., removing citations, URLs, signatures, names, code snippets, etc. (Biswas et al., 2020; Calefato et al., 2018; Imran et al., 2022; Imtiaz et al., 2018; Kaur et al., 2018; Klünder et al., 2020; Rousinopoulos et al., 2014; Sanei et al., 2021; Schneider et al., 2016; Tourani et al., 2014)). Such preprocessings (when applied prior to human annotation) could also influence humans' perceptions. Consequently, our annotation study shall be re-run while applying different text preprocessings (e.g., transforming all-caps text elements to lower-case text or replacing swear words), to find out whether and how these text elements affect human perception in a technical setting and whether these elements are actual causes for disagreement. Although such text preprocessings are not applied when developers receive e-mails in real-world scenarios, applying text preprocessings for research purposes would help increase the internal validity of human annotation studies in a technical context and obtain a better understanding of cause and effect of different text elements.

Beside analyzing the potential reasons for disagreement more thoroughly, also the development of measures on how to treat disagreement might be necessary, given that perfect agreement among humans is often not achievable. One option could be to adjust the sensitivity and specificity when aggregating annotation data into a single label. In what follows, we provide three primitive examples of how such an adjustment can be done when aggregating our annotation data, to demonstrate the potential pitfalls and difficulties that can occur during aggregation. We call them *potential ground truths* (GTs) for the aggressiveness of an e-mail that are determined via different aggregations of the human annotation data:[19]

- *GT1*: An e-mail is aggressive if the mode of its annotations is aggressive (considering a tie as non-aggressive, and excluding unsure annotations if there are ≥ 4 sure annotations for an e-mail).
- *GT2*: An e-mail is aggressive if, at least, one person has labeled it as aggressive (excluding all unsure annotations).
- *GT3*: Only e-mails where all but one persons agreed on the same binary label (excluding all unsure annotations) are considered.

While GT1 uses a measure of central tendency, GT3 uses only the subset of e-mails for which the annotators agreed on the binary label. Although GT3 seems to be most reliable, as it only considers e-mails with human agreement, it could potentially ignore a substantial amount of data. This is what has happened in our annotation study: There was substantial disagreement on 92 out of 720 e-mails. While, in some settings, ignoring disagreement might be justifiable, the controversial subset of the data could contain important samples that are of particular interest, precisely because of the disagreement (see Section 2). Thus, it quickly becomes obvious that all three examples (GT1, GT2, and GT3) are not reliable for our study, either because of the generally low agreement or because of ignoring e-mails with disagreement. With providing these three unreliable examples, we aim at stimulating further thought; future work shall try to develop a more reliable aggregation method that takes the level of disagreement into account. In this context, we also would like to put forward another thought: It might not be possible to develop more reliable aggregation methods without reconsidering or refining established inter-rater agreement measures. As our explorative experiment shows, achieving perfect agreement and at the same time representing all the diverse perceptions that humans can take might potentially be (close to) infeasible. Consequently, a new way of defining data reliability might be necessary in the context of highly subjective human-annotation studies.

Also the granularity of text snippets on which the annotation is performed should be subject to future investigations: While we decided to let the annotators label each individual e-mail as a whole, as the perception of an e-mail in its entirety is crucial for how developers react, it might be insightful to label individual paragraphs or sentences separately, for two reasons: On the one hand, paragraph- or sentence-level annotations might reveal which parts of an e-mail are perceived as aggressive and which parts lead to disagreement, this way being able to narrow down the potential reasons for disagreement more precisely. On the other hand, when comparing paragraph- or sentence-level annotations to the e-mail-level annotations, it could be investigated how many parts (i.e.,

[19] For demonstration purposes, we used these three examples as potential ground truths for aggressiveness in an exploratory experiment for the evaluation of the results of four selected sentiment analysis tools, using our dataset of 720 e-mails. This way, we show how the selection of the ground truth biases the comparison of human annotation data and tool results. We provide further details and results of this exploratory experiment on our supplementary website.

which fraction of an e-mail's paragraphs or sentences) are necessary to be perceived as aggressive in order to perceive the complete e-mail as aggressive. Such investigations could contribute toward an understanding of the amount or intensity of aggressiveness that might be tolerable in a healthy and welcoming project atmosphere in software-development projects. For instance, it may make a difference whether only a single mildly aggressive sentence or a single swear word is embedded in a mainly neutral or friendly e-mail or whether multiple aggressive instances are present in the e-mail. Also the position of aggressive content within the e-mail (e.g., at the beginning, in the middle, or at the end) could make a difference in how it is perceived. Consequently, it could be possible to develop continuous-scale metrics that assign different weights to different parts of an e-mail, depending on how likely it is that a specific part of an e-mail is perceived as aggressive. Such weights, in turn, could be used to develop continuous-scale ratings that account for the diverse and subjective human annotations instead of assigning a binary or discrete label to it. However, this is a topic on its own that raises additional research questions and, thus, is out of the scope of this paper.

Furthermore, it could also be interesting to derive different types of aggressiveness from the different potential causes of disagreement and analyze whether software developers react differently to such different types of aggressiveness. Nevertheless, additional studies are necessary to confirm these possible causes to be able to derive such types afterwards.

Beside deriving different types of aggressiveness, it could be useful to investigate whether and how different personality traits (Goldberg, 1993) might lead to different perceptions of aggressiveness. For example, it might be interesting to observe whether comparable personality types annotate more similarly or not.

Finally, it could also be beneficial for future investigations on developer perception to take advantage of arising cutting-edge technologies, such as large language models, which might be capable of identifying aggressive language. On the one hand, the usage of such tools could be helpful to obtain new insights into which language elements might cause aggressive or ambiguous perception in a fast and efficient way. On the other hand, using such technologies cannot reliably mimic human perceptions, given the diversity of human perceptions that we have observed in our annotation study. Therefore, dedicated empirical studies are necessary to determine whether and which configurations of cutting-edge technologies are suitable for application in the context of this research. Consequently, the investigation of such technologies in the context of sentiment analysis in the software-engineering domain opens a completely new research direction that goes along with additional caveats and methodological challenges.

8. Conclusion

As communication among software developers is a substantial part of OSS projects, conversational tone and aggressiveness can have an influence on developer participation. Initially, we aimed at investigating the effect of aggressive language in the LINUX kernel project. However, after conducting a human annotation study on 720 e-mails from the LKML with multiple annotators, we noticed that human agreement on aggressiveness is generally low and, therefore, the results of sentiment analysis tools (which are often trained on human-annotated data) can also not be reliable. Consequently, we conducted manual, qualitative investigations to understand why humans disagree. Although we identified potential causes for disagreement, we did not find a general theme beside the fact that different individuals may perceive aggressiveness differently. Our findings suggest that research in the software-engineering domain needs to differentiate between specific forms of aggressiveness which can be identified with less ambiguity and depend less on the personality and context of the person rating a text. Nevertheless, it is not sufficient to rely on aggregated measures of human annotations, as individual perceptions, particularly, matter when studying the effect of aggressive language in OSS projects. With publishing the experience we had throughout our study, we want to call attention to the methodological challenge in this research field, which should become an important part of future research.

Acknowledgments

We thank Tina Schuh and Barbara Eckl-Ganser for their support in preparing and conducting the annotation study and for fruitful discussions in early stages of this study. Furthermore, we thank the annotators for participating in our annotation study. This work was supported by the German Research Foundation (AP 206/14-1) as well as the Bavarian State Ministry of Education, Science, and the Arts in the framework of the Center Digitisation.Bavaria (ZD.B).

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

A. Appendix to Our Literature Review on Sentiment Analysis in Software Engineering

During our literature review (see Section 3), we collected more than 30 sentiment analysis tools that

Table 5
Papers related to sentiment analysis for software engineering, grouped into two different categories.

Category (# Papers)	Papers
Tool Development and Tool Evaluation (69)	Ahmed et al. (2017); Batra et al. (2021); Biswas et al. (2020, 2019); Blaz and Becker (2016); Bleyl and Buxton (2022); Cabrera-Diego et al. (2020); Cagnoni et al. (2020); Calefato et al. (2018, 2017, 2019); Cassee et al. (2022); Chen et al. (2019, 2021); Cheriyan et al. (2021); Ding et al. (2018); Efstathiou et al. (2018); Ferreira et al. (2021, 2024); Fucci et al. (2021); Gachechiladze et al. (2017); Herrmann et al. (2022); Imran et al. (2022); Imtiaz et al. (2018); Islam et al. (2019); Islam and Zibran (2017a,b, 2018a,b,d); Jongeling et al. (2015, 2017); Kadhar and Kumar (2022); Kaur et al. (2018); Klünder et al. (2020); Lin et al. (2019, 2018); Maipradit et al. (2019); Mansoor et al. (2021); Mäntylä et al. (2017); von der Mosel et al. (2023); Mula et al. (2022); Murgia et al. (2018); Neupane et al. (2019); Novielli et al. (2020, 2015, 2021, 2018b); Obaidi et al. (2022a); Park and Sharif (2021); Prenner and Robbes (2022); Qiu et al. (2022); Raman et al. (2020); Robbes and Janes (2019); Sarker et al. (2023a, 2020, 2023b); Serva et al. (2015); Shen et al. (2019); Sun et al. (2021, 2022); Uddin et al. (2022b); Uddin and Khomh (2021); Venigalla and Chimalakonda (2021a); Wang (2019); Werder and Brinkkemper (2018); Wu et al. (2021); Zhang et al. (2021, 2020)
Tool Application and Tool Usage (108)*	Ahasanuzzaman et al. (2018, 2020); Alesinloye et al. (2019); Almarimi et al. (2023); Assavakamhaenghan et al. (2023); Batoun et al. (2023); Brisson et al. (2020); Calefato et al. (2015); Chatterjee et al. (2021); Cheruvelil and da Silva (2019); Chouchen et al. (2021); Claes and Mäntylä (2020); Claes et al. (2018); Cohen (2021); da Cruz et al. (2016); Dao and Yang (2021); Destefanis et al. (2018, 2016); El Asri et al. (2019); Ferreira et al. (2022, 2019a,b,c); Freira et al. (2018); Gao et al. (2022); Garcia et al. (2013); Goyal and Sardana (2017); Graßl and Fraser (2022); Guzman (2013); Guzman et al. (2017, 2014); Guzman and Bruegge (2013); Hata et al. (2022); Herrmann and Klünder (2021); Huang et al. (2021); Huq et al. (2019, 2020); Imtiaz et al. (2019); Islam and Zibran (2016, 2018c); Jurado and Rodriguez (2015); Kaur et al. (2022); Kritikos et al. (2020); Kumar et al. (2022); Kuutila et al. (2020); Lanovaz and Adams (2019); Li et al. (2020, 2021); Licorish and MacDonell (2014, 2018); Madampe et al. (2020); Mahbub et al. (2021); Mäntylä et al. (2016); Marshall et al. (2016); Miller et al. (2022); Morales-Ramirez et al. (2019); Mostafa and Abd Elghany (2018); Munaiah et al. (2017); Murgia et al. (2014); Novielli et al. (2014, 2018a); Ortu et al. (2015a, 2016a, 2015b, 2018, 2019, 2016b); Patnaik and Padhy (2022); Patwardhan (2017); Paul et al. (2019); Pletea et al. (2014); Quintanilla Portugal and Sampaio do Prado Leite (2018); Rahman et al. (2015); Ramay et al. (2019); Robe et al. (2022); Robinson et al. (2016); Rong et al. (2022); Rousinopoulos et al. (2014); Sanei et al. (2021); Sapkota et al. (2019); Sarker et al. (2019); Sayago-Heredia et al. (2022a,b); Schroth et al. (2022); Sengupta and Haythornthwaite (2020); Singh and Singh (2017); Sinha et al. (2016); Skriptsova et al. (2019); Sokolovsky et al. (2021); Souza and Silva (2017); Swillus and Zaidman (2023); Tourani and Adams (2016); Tourani et al. (2014); Uddin et al. (2022a, 2020, 2021); Umer et al. (2020); Valdez et al. (2020); Venigalla and Chimalakonda (2021b); Wang et al. (2023); Werder (2018); Werner et al. (2019, 2018); Yang et al. (2017a,b, 2018); Yin et al. (2023); Zhang and Hou (2013)

* In this category, we also included papers that only used human annotation (manual labeling) as a sentiment analysis tool. This concerns the following papers: Batoun et al. (2023); Chouchen et al. (2021); Ferreira et al. (2022); Marshall et al. (2016); Murgia et al. (2014); Robe et al. (2022); Sengupta and Haythornthwaite (2020); Uddin et al. (2022a).

have been developed for the software-engineering domain and 69 papers that addressed tool development and tool evaluation. The remaining 108 papers that we considered relevant reported on the application and usage of such tools for the purpose of answering specific research questions in the software-engineering domain. An overview of which paper belongs to which of the two categories can be found in Table 5, and an overview of the different venues at which the papers have been published can be found in Table 6. In the following, we provide a brief overview of the collected tools and of the results researchers obtained from using sentiment analysis tools.

A.1. Tool Development and Tool Evaluation

As a side result of our literature review, we found that many researchers developed their own sentiment analysis tools specifically tuned to the software-engineering domain, based on and evaluated on manually labeled datasets such as code review comments, issue comments, ticket systems, or StackOverflow posts. Whereas some researchers use self-created lexicons or dictionaries containing IT vocabulary (Blaz and Becker, 2016; Mäntylä et al., 2017), others develop their own tools and train them on previously manually labeled data, or evaluate them against existing tools and show that their tuning for software-engineering texts significantly improves classification accuracy. In the following, we briefly list the tools and their related papers that we found through our literature review:[20] SentiCR (Ahmed et al., 2017), EmoTxt (Calefato et al., 2017), SentiStrength-SE (Islam and Zibran, 2017a,b, 2018a,d), DEVA (Islam and Zibran, 2018b), Stanford CoreNLP SO (Lin et al., 2018), MEME (Werder and Brinkkemper, 2018), ESEM-E (Murgia et al., 2018), Senti4SD (Calefato et al., 2018), SentiSW (Ding et al., 2018), word2vec (Efstathiou et al., 2018), EMTk (Calefato et al., 2019), POME (Lin et al., 2019), MarValous (Islam et al., 2019), RNN4SentiSE (Biswas et al., 2019), BERT4SentiSE (Biswas et al., 2020), BERT-FT (Wu

[20] We provide further information about all these papers as well as the corresponding tools and approaches on our supplementary website.

Table 6
Venues containing papers related to sentiment analysis for software engineering.

Venue[5]	# Papers	Papers
MSR	20	Biswas et al. (2019); Blaz and Becker (2016); Calefato et al. (2015); Claes and Mäntylä (2020); Efstathiou et al. (2018); Ferreira et al. (2022); Fucci et al. (2021); Guzman et al. (2014); Islam and Zibran (2017b); Mäntylä et al. (2016); Mäntylä et al. (2017); Murgia et al. (2014); Novielli et al. (2020, 2018a,b); Ortu et al. (2015a, 2016b); Pletea et al. (2014); Sinha et al. (2016); Souza and Silva (2017)
SEmotion	12	Calefato et al. (2019); Cheruvelil and da Silva (2019); Destefanis et al. (2018); Ding et al. (2018); Ferreira et al. (2019a); Imtiaz et al. (2018); Mansoor et al. (2021); Marshall et al. (2016); Ortu et al. (2019); Park and Sharif (2021); Werder (2018); Werder and Brinkkemper (2018)
EMSE	10	Ahasanuzzaman et al. (2020); Assavakamhaenghan et al. (2023); Calefato et al. (2018); Cassee et al. (2022); Hata et al. (2022); Jongeling et al. (2017); Murgia et al. (2018); Novielli et al. (2021); Uddin et al. (2022a); Wang et al. (2023)
ICSE	7	Chatterjee et al. (2021); Imtiaz et al. (2019); Lin et al. (2019, 2018); Miller et al. (2022); Sarker et al. (2019); Yin et al. (2023)
SANER	7	Ahasanuzzaman et al. (2018); Brisson et al. (2020); Chouchen et al. (2021); Huq et al. (2020); Islam and Zibran (2018a); Paul et al. (2019); Tourani and Adams (2016)
ESEC/FSE	5	Chen et al. (2019); Cohen (2021); Guzman and Bruegge (2013); Sarker et al. (2020); Venigalla and Chimalakonda (2021a)
JSS	5	Ferreira et al. (2024); Herrmann et al. (2022); Islam and Zibran (2018d); Jurado and Rodriguez (2015); Swillus and Zaidman (2023)
TOSEM	5	Batoun et al. (2023); Chen et al. (2021); Sarker et al. (2023b); Uddin et al. (2022b, 2021)
APSEC	4	Huq et al. (2019); Li et al. (2021); Sarker et al. (2020); Singh and Singh (2017)
ICSE-NIER	4	Gachechiladze et al. (2017); Madampe et al. (2020); Raman et al. (2020); Robbes and Janes (2019)
ICSME	4	Biswas et al. (2020); Jongeling et al. (2015); Neupane et al. (2019); Zhang et al. (2020)
IST	4	El Asri et al. (2019); Licorish and MacDonell (2018); Rong et al. (2022); Uddin et al. (2020)
ASE	3	Ahmed et al. (2017); Imran et al. (2022); Wang (2019)
ESEM	3	Claes et al. (2018); Islam and Zibran (2017a); Sarker et al. (2023a)
ICPC	3	Huang et al. (2021); Sun et al. (2021); Zhang and Hou (2013)
ICSE-SEIS	3	Graßl and Fraser (2022); Qiu et al. (2022); Venigalla and Chimalakonda (2021b)
IEEE Software	3	Lanovaz and Adams (2019); Maipradit et al. (2019); Werner et al. (2019)
SAC	3	Islam et al. (2019); Islam and Zibran (2018b); Yang et al. (2017b)
TSE	3	von der Mosel et al. (2023); Prenner and Robbes (2022); Uddin and Khomh (2021)
AffectRE	2	Quintanilla Portugal and Sampaio do Prado Leite (2018); Werner et al. (2018)
EISEJ	2	Goyal and Sardana (2017); Kaur et al. (2022)
HCSE	2	Klünder et al. (2020); Schroth et al. (2022)
HICSS	2	Robinson et al. (2016); Sengupta and Haythornthwaite (2020)
OpenSym	2	Alesinloye et al. (2019); Ferreira et al. (2019c)
PLOS ONE	2	Sapkota et al. (2019); Sokolovsky et al. (2021)
SEKE	2	Freira et al. (2018); Li et al. (2020)
SSE	2	Novielli et al. (2014, 2015)
XP	2	Ortu et al. (2016a, 2015b)
others*	51	Almarimi et al. (2023); Batra et al. (2021); Bleyl and Buxton (2022); Cabrera-Diego et al. (2020); Cagnoni et al. (2020); Calefato et al. (2017); Cheriyan et al. (2021); da Cruz et al. (2016); Dao and Yang (2021); Destefanis et al. (2016); Ferreira et al. (2021, 2019b); Gao et al. (2022); Garcia et al. (2013); Guzman (2013); Guzman et al. (2017); Herrmann and Klünder (2021); Islam and Zibran (2016, 2018c); Kadhar and Kumar (2022); Kaur et al. (2018); Kritikos et al. (2020); Kumar et al. (2022); Kuutila et al. (2020); Licorish and MacDonell (2014); Mahbub et al. (2021); Morales-Ramirez et al. (2019); Mostafa and Abd Elghany (2018); Mula et al. (2022); Munaiah et al. (2017); Obaidi et al. (2022a); Ortu et al. (2018); Patnaik and Padhy (2022); Patwardhan (2017); Rahman et al. (2015); Ramay et al. (2019); Rousinopoulos et al. (2014); Sanei et al. (2021); Sayago-Heredia et al. (2022a,b); Serva et al. (2015); Shen et al. (2019); Skriptsova et al. (2019); Sun et al. (2022); Tourani et al. (2014); Umer et al. (2020); Valdez et al. (2020); Wu et al. (2021); Yang et al. (2017a, 2018); Zhang et al. (2021)

* ACIIW, ACIS, APSECW, ASEW, BigData, CASCON, CGC, CHASE, CIC, CONISOFT, DASC, DEXA, DTGS, EASE, EDM, ENASE, ESSoS, HCI, HotStorage, I3E, ICACI, ICAT, ICCSAW, ICECA, ICEIS, ICITA, ICNGIoT, ICSESS, ICSEW, ICSS, ICT Express, IEEE Access, IJSEA, Information Systems, Internetware, Journal of Software: Evolution and Process, KBS, Mathematics, OSS, PeerJ Comp. Sci., PROFES, PROMISE, RE Journal, RESI, REW, SCAM, SEDE, SERA, Trans. Info. Syst., TRel, VISSOFT (1 paper per venue)

et al., 2021), OpinerDSO (Uddin and Khomh, 2021), SentiLog (Zhang et al., 2021), SESSION (Sun et al., 2021), EASTER (Sun et al., 2022), SentiSead (Uddin et al., 2022b), StackOBERTflow (Prenner and Robbes, 2022), seBERT (von der Mosel et al., 2023), and different fine-tuned transformer models (e.g., Zhang et al., 2020), BERT-based language models (e.g., Batra et al., 2021; Biswas et al., 2020; Bleyl and Buxton, 2022; Prenner and Robbes, 2022; Wu et al., 2021), or other machine-learning models (e.g., Klünder et al., 2020; Maipradit et al., 2019). Some tools even have been developed for specific aspects: For example, Gachechiladze et al. (2017) developed a tool to identify anger direction (i.e., whether anger is

directed against the commenters themselves, against other people, or against objects). Sarker et al. (2020, 2023b) developed ToxiCR to identify toxic comments. In addition, Sarker et al. (2023a) developed ToxiSpanSE, which is even able to detect which parts of a toxic comment are causing the toxicity. Many researchers combined and compared various tools and built their own toxicity detection tools based on existing approaches (Cheriyan et al., 2021; Qiu et al., 2022; Raman et al., 2020; Sayago-Heredia et al., 2022a). As another special aspect, the tools SentiMoji (Chen et al., 2019) and StackEmo (Venigalla and Chimalakonda, 2021a) especially deal with emojis in software-engineering-related texts. To investigate how developers' emotions change over time, Neupane et al. (2019) developed the tool EmoD, which uses a combination of already existing sentiment analysis tools. Similarly, Cagnoni et al. (2020) used multiple machine-learning algorithms to detect joy, love, surprise, fear, anger, and sadness in StackOverflow posts regarding different programming languages.

We conclude that the analysis of developers' sentiments is a highly relevant research topic since so many tools have been developed specifically for the software-engineering domain. However, the huge number of different approaches, datasets, and tools that we have identified during our literature review also indicates that existing approaches may not be accurate and reliable enough. This can also be seen from the high number of tool evaluation papers that we already have referenced in Section 3.1.1.

A.2. Tool Application and Tool Usage

In what follows, we provide an overview of relevant studies that aimed at answering specific research questions related to sentiment in the software-engineering domain by using sentiment analysis tools.[20] In this category, we also included papers that only used human annotation (manual labeling) as a sentiment analysis tool. Due to the concerns regarding tool reliability and human perception raised above, the results reported in the following have to be taken with a grain of salt. In addition to the selection of studies that we have already presented in Section 3.1.2, we now provide details on the remaining studies on tool application and tool usage that we have found in our literature review:

Research has shown that emotions and sentiment polarity are present in the communication channels of OSS projects (Ferreira et al., 2019c; Graßl and Fraser, 2022; Guzman and Bruegge, 2013; Jurado and Rodriguez, 2015; Murgia et al., 2014; Tourani et al., 2014), but usually only a small fraction of the communication expresses a positive or negative sentiment (Ferreira et al., 2019c; Hata et al., 2022; Sengupta and Haythornthwaite, 2020; Skriptsova et al., 2019; Valdez et al., 2020). Whereas most GitHub projects are neutral, there are 10% more projects with negative sentiment than projects with positive sentiment (Sinha et al., 2016). However, discussions on GitHub seem to contain more positive sentiment than on StackOverflow (Hata et al., 2022); nevertheless, avoiding negative attitude on StackOverflow increases one's chances to get an answer accepted (Calefato et al., 2015). The sentiment of developers in software projects is subject to constant change. Multiple studies have shown that the amount of positive sentiment in discussions decreases over time (Robinson et al., 2016; Rousinopoulos et al., 2014; Werder, 2018). Moreover, contributions of developers tend to become more emotional and longer during a project's lifetime (Guzman, 2013). This is corroborated by the empirical observation that positive and negative comments tend to be longer than neutral comments (Lanovaz and Adams, 2019). Ortu et al. (2018) found that communication styles change with the number of commits: Developers who only contributed one commit were more polite than developers who contributed regularly. Also, developers who have a significantly higher commenting activity than their peers often tend to an increased amount of negative sentiment (Sarker et al., 2019). By contrast, also people that never created an issue and never contributed to the source code but just comment on issues are less polite than others (Destefanis et al., 2018). In general, conversations among developers are more neutral compared to conversations between developers and users (Robe et al., 2022).

Multiple studies detect the sentiment in bug reports to predict the bug severity in order to automatically prioritize bug reports according to their severity (Dao and Yang, 2021; Ramay et al., 2019; Umer et al., 2020; Yang et al., 2017b, 2018). In a similar vein, Ahasanuzzaman et al. (2018, 2020) use sentiment information to classify whether a StackOverflow post describes an issue or not, and Werner et al. (2019, 2018) show that sentiment can be used to identify escalated support tickets.

Munaiah et al. (2017) investigated the sentiment in code reviews and found that the more emotional and the less complex the code changes are, the more likely a code review fails to notice vulnerabilities. Furthermore, Tourani and Adams (2016) showed that the more negative sentiment occurs in a code review, the more defect-prone the code changes are. In addition, pull requests that contain anger or sadness have a lower probability to be merged than pull requests that contain positive emotions (Ortu et al., 2019). According to El Asri et al. (2019), code reviews with negative comments also need more time to be addressed by the developer than code reviews with positive comments. In particular, newcomers react more emotional to code reviews than core members of a project do (El Asri et al., 2019; Skriptsova et al., 2019).

Sentiment analysis tools are also used to examine how sentiment is related to different programming languages (Guzman et al., 2014), a developer's gender (Imtiaz et al., 2019; Patwardhan, 2017; Paul et al., 2019), developer productivity (Kuutila et al., 2020; Licorish and MacDonell, 2014, 2018), the presence of bots (Gao et al., 2022), requirements changes (Madampe et al., 2020), code quality (Sayago-Heredia et al., 2022a,b), refactoring activities (Patnaik and Padhy, 2022; Singh and Singh, 2017), software testing (Swillus and Zaidman, 2023), a project's attractiveness (Brisson et al., 2020; Destefanis et al., 2016; Ortu et al., 2015b), community smells (Almarimi et al., 2023; Huang et al., 2021), and issue fixing time (Destefanis et al., 2016; Mäntylä et al., 2016; Ortu et al., 2015a,b, 2016b; Sanei et al., 2021; Yang et al., 2017a). Also, the role of emojis (Batoun et al., 2023; Claes et al., 2018; Rong et al., 2022; Wang et al., 2023), the guiltiness of game developers with respect to the negative effects of game addiction (Mostafa and Abd Elghany, 2018), and the technical information contained in tweets about software applications (Guzman et al., 2017) have been studied.

Sokolovsky et al. (2021) used sentiment analysis to predict software releases, since emotions change during the course of a release cycle (e.g., there is more negative sentiment in the days prior to a release) (Alesinloye et al., 2019; Ferreira et al., 2019b). Sentiment analysis is also used to automatically detect trust between developers (da Cruz et al., 2016; Sapkota et al., 2019). Other researchers investigated whether dependencies between non-functional software requirements can be derived from the sentiments in issue comments (Quintanilla Portugal and Sampaio do Prado Leite, 2018). Zhang and Hou (2013) extracted problematic API features from forum discussions based on negative sentiment. Moreover, Uddin et al. (2020, 2021) identified developers' sentiment in StackOverflow comments and use this information to generate API documentation therefrom. Especially during the first months of the COVID-19 pandemic, developers often complained about missing documentation, which was reflected in negative sentiment (Uddin et al., 2022a). To explore future usage scenarios of sentiment analysis in software engineering, Schroth et al. (2022) tried out the concept of "realtime sentiment analysis", (i.e., visualizing sentiment scores to developers while they type a message). While some developers considered this useful, others voiced misgivings due to being observed while typing the message.

To be able to identify sentiment in oral developer communication, Herrmann and Klünder (2021) used speech recognition to transcribe the oral conversations and applied sentiment analysis tools afterwards. To detect "speech acts" (i.e., communication that should affect other people's believes or behavior), also sentiment analysis tools can be used to identify positive or negative opinions (Morales-Ramirez et al., 2019).

As already described in Section 3.1.2, Ferreira et al. (2019a) assessed on the LKML whether the maintainers' sentiment changed after Linus Torvalds's temporary break and did not find any significant changes. Ferreira et al. (2021) investigated potential causes for incivility in 1 545 e-mails from the LKML and focused on understanding the communication between the developers.

All in all, the results of all the above mentioned studies are very interesting and diverse (sometimes even contradicting), but due to the concerns raised in our study, one cannot fully rely on these results. This is why we aimed at evaluating sentiment analysis tools for our purpose before using them, which is why we started with our human annotation study.

References

Ahasanuzzaman, M., Asaduzzaman, M., Roy, C.K., Schneider, K.A., 2018. Classifying Stack Overflow Posts on API Issues, in: Int. Conf. Software Analysis, Evolution, and Reengineering (SANER), IEEE. pp. 244–254.

Ahasanuzzaman, M., Asaduzzaman, M., Roy, C.K., Schneider, K.A., 2020. CAPS: A Supervised Technique for Classifying StackOverflow Posts Concerning API Issues. Empirical Software Engineering (EMSE) 25, 1493–1532.

Ahmed, T., Bosu, A., Iqbal, A., Rahimi, S., 2017. SentiCR: A Customized Sentiment Analysis Tool for Code Review Interactions, in: Proc. Int. Conf. Automated Software Engineering (ASE), IEEE. pp. 106–111.

Alami, A., Leavitt Cohn, M., Wąsowski, A., 2019. Why Does Code Review Work for Open Source Software Communities?, in: Proc. Int. Conf. Software Engineering (ICSE), IEEE. pp. 1073–1083.

Alesinloye, J.A., Groarke, E., Babu, J., Srinivasan, S., Curran, G., Dennehy, D., 2019. Sentiment Analysis of Open Source Software Community Mailing List: A Preliminary Analysis, in: Proc. Int. Sympos. Open Collaboration (OpenSym), ACM. pp. 21:1–21:5.

Almarimi, N., Ouni, A., Chouchen, M., Mkaouer, M.W., 2023. Improving the Detection of Community Smells Through Socio-Technical and Sentiment Analysis. Journal of Software: Evolution and Process 35, e2505.

Ames, D.L., Fiske, S.T., 2013. Intentional Harms Are Worse, Even When They're Not. Psychological Science 24, 1755–1762.

Assavakamhaenghan, N., Wattanakriengkrai, S., Shimada, N., Kula, R.G., Ishio, T., Matsumoto, K., 2023. Does the First Response Matter for Future Contributions? A Study of First Contributions. Empirical Software Engineering (EMSE) 28, 75.

Babchuk, W.A., 1996. Glaser or Strauss? Grounded Theory and Adult Education, in: Proc. Annual Midwest Research-to-Practice Conf. in Adult, Continuing, and Community Education, University of Nebraska-Lincoln. pp. 1–6.

Baron, R.A., Richardson, D.R., 1994. Human Aggression. Studies in Natural Language Processing. 2 ed., Springer.

Batoun, M.A., Yung, K.L., Tian, Y., Sayagh, M., 2023. An Empirical Study on GitHub Pull Requests' Reactions. ACM Transactions on Software Engineering and Methodology (TOSEM) 32, 146.

Batra, H., Punn, N.S., Sonbhadra, S.K., Agarwal, S., 2021. BERT-Based Sentiment Analysis: A Software Engineering Perspective, in: Database and Expert Systems Applications (DEXA), Springer. pp. 138–148.

Berkowitz, L., 1993. Aggression: Its Causes, Consequences, and Control. McGraw-Hill Book Company.

Bhat, M.M., Hosseini, S., Awadallah, A.H., Bennett, P., Li, W., 2021. Say 'YES' to Positivity: Detecting Toxic Language in Workplace Communications, in: Findings of the Association for Computational Linguistics: EMNLP, ACL. pp. 2017–2029.

Biswas, E., Karabulut, M.E., Pollock, L.L., Vijay-Shanker, K., 2020. Achieving Reliable Sentiment Analysis in the Software Engineering Domain Using BERT, in: Proc. Int. Conf. Software Maintenance and Evolution (ICSME), IEEE. pp. 162–173.

Biswas, E., Vijay-Shanker, K., Pollock, L.L., 2019. Exploring Word Embedding Techniques to Improve Sentiment Analysis of Software Engineering Texts, in: Proc. Int. Workshop on Mining Software Repositories (MSR), IEEE. pp. 68–78.

Blaz, C.C.A., Becker, K., 2016. Sentiment Analysis in Tickets for IT Support, in: Proc. Int. Workshop on Mining Software Repositories (MSR), ACM. pp. 235–246.

Bleyl, D., Buxton, E.K., 2022. Emotion Recognition on StackOverflow Posts Using BERT, in: Int. Conf. Big Data (Big Data), IEEE. pp. 5881–5885.

Brisson, S., Noei, E., Lyons, K., 2020. We Are Family: Analyzing Communication in GitHub Software Repositories and Their Forks, in: Int. Conf. Software Analysis, Evolution, and Reengineering (SANER), IEEE. pp. 59–69.

Burroughs, S.M., James, L.R., 2005. Advancing the Assessment of Dispositional Aggressiveness Through Conditional Reasoning, in: Counterproductive Work Behavior: Investigations of Actors and Targets. American Psychological Association, pp. 127–150.

Buss, A.H., 1971. Aggression Pays. The Control of Aggression and Violence: Cognitive and Physiological Factors , 7–18.

Cabrera-Diego, L.A., Bessis, N., Korkontzelos, I., 2020. Classifying Emotions in Stack Overflow and JIRA Using a Multi-Label Approach. Knowledge-Based Systems (KBS) 195, 105633.

Cagnoni, S., Cozzini, L., Lombardo, G., Mordonini, M., Poggi, A., Tomaiuolo, M., 2020. Emotion-Based Analysis of Programming Languages on StackOverflow. ICT Express 6, 238–242.

Calefato, F., Lanubile, F., Maiorano, F., Novielli, N., 2018. Sentiment Polarity Detection for Software Development. Empirical Software Engineering (EMSE) 23, 1352–1382.

Calefato, F., Lanubile, F., Marasciulo, M.C., Novielli, N., 2015. Mining Successful Answers in Stack Overflow, in: Proc. Int. Workshop on Mining Software Repositories (MSR), IEEE. pp. 430–433.

Calefato, F., Lanubile, F., Novielli, N., 2017. EmoTxt: A Toolkit for Emotion Recognition from Text, in: Int. Conf. Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW), IEEE. pp. 79–80.

Calefato, F., Lanubile, F., Novielli, N., Quaranta, L., 2019. EMTk – The Emotion Mining Toolkit, in: Proc. Int. Workshop on Emotion Awareness in Software Engineering (SEmotion), IEEE. pp. 34–37.

Canfora, G., Di Penta, M., Oliveto, R., Panichella, S., 2012. Who is Going to Mentor Newcomers in Open Source Projects?, in: Proc. Int. Sympos. Foundations of Software Engineering (FSE), ACM. pp. 1–11.

Cassee, N., Zampetti, F., Novielli, N., Serebrenik, A., Di Penta, M., 2022. Self-Admitted Technical Debt and Comments' Polarity: An Empirical Study. Empirical Software Engineering (EMSE) 27, 139.

Cataldo, M., Herbsleb, J.D., 2013. Coordination Breakdowns and Their Impact on Development Productivity and Software Failures. IEEE Transactions on Software Engineering (TSE) 39, 343–360.

Chatterjee, P., Damevski, K., Pollock, L., 2021. Automatic Extraction of Opinion-Based Q&A from Online Developer Chats, in: Proc. Int. Conf. Software Engineering (ICSE), IEEE. pp. 1260–1272.

Chen, Z., Cao, Y., Lu, X., Mei, Q., Liu, X., 2019. SEntiMoji: An Emoji-Powered Learning Approach for Sentiment Analysis in Software Engineering, in: Proc. Europ. Software Engineering Conf. and the Int. Sympos. Foundations of Software Engineering (ESEC/FSE), ACM. pp. 841–852.

Chen, Z., Cao, Y., Yao, H., Lu, X., Peng, X., Mei, H., Liu, X., 2021. Emoji-Powered Sentiment and Emotion Detection from Software Developers' Communication Data. ACM Transactions on Software Engineering and Methodology (TOSEM) 30, 18.

Cheriyan, J., Savarimuthu, B.T.R., Cranefield, S., 2021. Towards Offensive Language Detection and Reduction in Four Software Engineering Communities, in: Evaluation and Assessment in Software Engineering (EASE), ACM. pp. 254–259.

Cheruvelil, J., da Silva, B.C., 2019. Developers' Sentiment and Issue Reopening, in: Proc. Int. Workshop on Emotion Awareness in Software Engineering (SEmotion), IEEE. pp. 29–33.

Chouchen, M., Ouni, A., Kula, R.G., Wang, D., Thongtanunam, P., Mkaouer, M.W., Matsumoto, K., 2021. Anti-Patterns in Modern Code Review: Symptoms and Prevalence, in: Int. Conf. Software Analysis, Evolution, and Reengineering (SANER), IEEE. pp. 531–535.

Claes, M., Mäntylä, M.V., 2020. 20-MAD: 20 Years of Issues and Commits of Mozilla and Apache Development, in: Proc. Int. Workshop on Mining Software Repositories (MSR), ACM. pp. 503–507.

Claes, M., Mäntylä, M.V., Farooq, U., 2018. On the Use of Emoticons in Open Source Software Development, in: Proc. Int. Sympos. Empirical Software Engineering and Measurement (ESEM), ACM. pp. 1–4.

Cohen, S., 2021. Contextualizing Toxicity in Open Source: A Qualitative Study, in: Proc. Europ. Software Engineering Conf. and the Int. Sympos. Foundations of Software Engineering (ESEC/FSE), ACM. pp. 1669–1671.

da Cruz, G.A.M., Moriya-Huzita, E.H., Feltrim, V.D., 2016. Estimating Trust in Virtual Teams, in: Int. Conf. Enterprise Information Systems (ICEIS), SCITEPRESS – Science and Technology Publications. pp. 464–471.

D'Andrea, A., Ferri, F., Grifoni, P., Guzzo, T., 2015. Approaches, Tools and Applications for Sentiment Analysis Implementation. International Journal of Computer Applications 125, 26–33.

Dao, A.H., Yang, C.Z., 2021. Severity Prediction for Bug Reports Using Multi-Aspect Features: A Deep Learning Approach. Mathematics 9, 1644.

Del Bosque, L.P., Garza, S.E., 2014. Aggressive Text Detection for Cyberbullying, in: Proc. Mexcian Int. Conf. Human-Inspired Computing and Its Applications (MICAI), Springer. pp. 221–232.

Destefanis, G., Ortu, M., Bowes, D., Marchesi, M., Tonelli, R., 2018. On Measuring Affects of GitHub Issues' Commenters, in: Proc. Int. Workshop on Emotion Awareness in Software Engineering (SEmotion), ACM. pp. 14–19.

Destefanis, G., Ortu, M., Counsell, S., Swift, S., Marchesi, M., Tonelli, R., 2016. Software Development: Do Good Manners Matter? PeerJ Computer Science 2, e73.

Díaz, J., Pérez, J., Gallardo, C., González-Prieto, Á., 2023. Applying Inter-Rater Reliability and Agreement in Collaborative Grounded Theory Studies in Software Engineering. Journal of Systems and Software (JSS) 195, 111520.

Ding, J., Sun, H., Wang, X., Liu, X., 2018. Entity-Level Sentiment Analysis of Issue Comments, in: Proc. Int. Workshop on Emotion Awareness in Software Engineering (SEmotion), ACM. pp. 7–13.

Efstathiou, V., Chatzilenas, C., Spinellis, D., 2018. Word Embeddings for the Software Engineering Domain, in: Proc. Int. Workshop on Mining Software Repositories (MSR), ACM. pp. 38–41.

Egelman, C.D., Murphy-Hill, E., Kammer, E., Hodges, M.M., Green, C., Jaspan, C., Lin, J., 2020. Predicting Developers' Negative Feelings about Code Review, in: Proc. Int. Conf. Software Engineering (ICSE), ACM. pp. 174–185.

Ehrlich, K., Cataldo, M., 2012. All-for-One and One-for-All? A Multi-Level Analysis of Communication Patterns and Individual Performance in Geographically Distributed Software Development, in: Proc. Int. Conf. Computer-Supported Cooperative Work (CSCW), ACM. pp. 945–954.

El Asri, I., Kerzazi, N., Uddin, G., Khomh, F., Idrissi, M.A.J., 2019. An Empirical Study of Sentiments in Code Reviews. Information and Software Technology (IST) 114, 37–54.

Ferreira, I., Adams, B., Cheng, J., 2022. How Heated Is It? Understanding GitHub Locked Issues, in: Proc. Int. Workshop on Mining Software Repositories (MSR), ACM. pp. 309–320.

Ferreira, I., Cheng, J., Adams, B., 2021. The "Shut the F**k up" Phenomenon: Characterizing Incivility in Open Source Code Review Discussions. Proceedings of the ACM on Human-Computer Interaction (HCI) 5, 353.

Ferreira, I., Rafiq, A., Cheng, J., 2024. Incivility Detection in Open Source Code Review and Issue Discussions. Journal of Systems and Software (JSS) 209, 111935.

Ferreira, I., Stewart, K., German, D., Adams, B., 2019a. A Longitudinal Study on the Maintainers' Sentiment of a Large Scale Open Source Ecosystem, in: Proc. Int. Workshop on Emotion Awareness in Software Engineering (SEmotion), IEEE. pp. 17–22.

Ferreira, J., Dennehy, D., Babu, J., Conboy, K., 2019b. Winning of Hearts and Minds: Integrating Sentiment Analytics into the Analysis of Contradictions, in: Conf. e-Business, e-Services, e-Society (I3E), Springer. pp. 392–403.

Ferreira, J., Glynn, M., Hunt, D., Babu, J., Dennehy, D., Conboy, K., 2019c. Sentiment Analysis of Open Source Communities: An Exploratory Study, in: Proc. Int. Sympos. Open Collaboration (OpenSym), ACM. pp. 20:1–20:5.

Fortuna, P., Nunes, S., 2019. A Survey on Automatic Detection of Hate Speech in Text. ACM Computing Surveys 51, 85.

Freira, M., Caetano, J., Oliveira, J., Marques-Neto, H., 2018. Analyzing the Impact of Feedback in GitHub on the Software Developer's Mood, in: Proc. Int. Conf. Software Engineering & Knowledge Engineering (SEKE), KSI Research Inc. and Knowledge Systems Institute Graduate School. pp. 445–444.

Fucci, G., Cassee, N., Zampetti, F., Novielli, N., Serebrenik, A., Di Penta, M., 2021. Waiting Around or Job Half-Done? Sentiment in Self-Admitted Technical Debt, in: Proc. Int. Workshop on Mining Software Repositories (MSR), IEEE. pp. 403–414.

Gachechiladze, D., Lanubile, F., Novielli, N., Serebrenik, A., 2017. Anger and Its Direction in Collaborative Software Development, in: Proc. Int. Conf. Software Engineering: New Ideas and Emerging Results (ICSE-NIER), IEEE. pp. 11–14.

Gao, A., Chen, S., Wang, T., Deng, J., 2022. Understanding the Impact of Bots on Developers Sentiment and Project Progress, in: Int. Conf. Software Engineering and Service Science (ICSESS), IEEE. pp. 93–96.

Garcia, D., Zanetti, M.S., Schweitzer, F., 2013. The Role of Emotions in Contributors Activity: A Case Study on the GENTOO Community, in: Proc. Int. Conf. Cloud and Green Computing (CGC), IEEE. pp. 410–417.

Garg, T., Masud, S., Suresh, T., Chakraborty, T., 2023. Handling Bias in Toxic Speech Detection: A Survey. ACM Computing Surveys 55, 264.

Goldberg, L.R., 1993. The Structure of Phenotypic Personality Traits. American Psychologist 48, 26–34.

Goulding, C., 2002. Grounded Theory: A Practical Guide for Management, Business and Market Researchers. Sage.

Goyal, A., Sardana, N., 2017. NRFixer: Sentiment Based Model for Predicting the Fixability of Non-Reproducible Bugs. e-Informatica Software Engineering Journal (EISEJ) 11, 103–116.

Graßl, I., Fraser, G., 2022. Scratch as Social Network: Topic Modeling and Sentiment Analysis in Scratch Projects, in: Proc. Int. Conf. Software Engineering: Software Engineering in Society (ICSE-SEIS), ACM. pp. 143–148.

Grinter, R.E., Herbsleb, J.D., Perry, D.E., 1999. The Geography of Coordination: Dealing with Distance in R&D Work, in: Proc. Int. Conf. Supporting Group Work (GROUP), ACM. pp. 306–315.

Guzman, E., 2013. Visualizing Emotions in Software Development Projects, in: Working Conference on Software Visualization (VISSOFT), IEEE. pp. 1–4.

Guzman, E., Alkadhi, R., Seyff, N., 2017. An Exploratory Study of Twitter Messages about Software Applications. Requirements Engineering (RE) 22, 387–412.

Guzman, E., Azócar, D., Li, Y., 2014. Sentiment Analysis of Commit Comments in GitHub: An Empirical Study, in: Proc. Int. Workshop on Mining Software Repositories (MSR), ACM. pp. 352–355.

Guzman, E., Bruegge, B., 2013. Towards Emotional Awareness in Software Development Teams, in: Proc. Europ. Software Engineering Conf. and the Int. Sympos. Foundations of Software Engineering (ESEC/FSE), ACM. pp. 671–674.

Hamilton, M., 2012. Verbal Aggression: Understanding the Psychological Antecedents and Social Consequences. Journal of Language and Social Psychology 31, 5–12.

Hata, H., Novielli, N., Baltes, S., Kula, R.G., Treude, C., 2022. GitHub Discussions: An Exploratory Study of Early Adoption. Empirical Software Engineering (EMSE) 27, 3.

Herbsleb, J.D., Mockus, A., Roberts, J.A., 2006. Collaboration in Software Engineering Projects: A Theory of Coordination, in: Proc. Int. Conf. Information Systems (ICIS), Association for Information Systems. pp. 553–568.

Herrmann, M., Klünder, J., 2021. From Textual to Verbal Communication: Towards Applying Sentiment Analysis to a Software Project Meeting, in: Int. Requirements Engineering Conf. Workshops (REW), IEEE. pp. 371–376.

Herrmann, M., Obaidi, M., Chazette, L., Klünder, J., 2022. On the Subjectivity of Emotions in Software Projects: How Reliable Are Pre-labeled Data Sets for Sentiment Analysis? Journal of Systems and Software (JSS) 193, 111448.

Holm, O., 1980. Attribution of Aggressiveness and Judgment of Behavior as Right or Wrong. Umeå Universitet.

Huang, Z., Shao, Z., Fan, G., Gao, J., Zhou, Z., Yang, K., Yang, X., 2021. Predicting Community Smells' Occurrence on Individual Developers by Sentiments, in: Proc. Int. Conf. Program Comprehension (ICPC), IEEE. pp. 230–241.

Huq, S.F., Sadiq, A.Z., Sakib, K., 2019. Understanding the Effect of Developer Sentiment on Fix-Inducing Changes: An Exploratory Study on GitHub Pull Requests, in: Proc. Asia-Pacific Software Engineering Conf. (APSEC), IEEE. pp. 514–521.

Huq, S.F., Sadiq, A.Z., Sakib, K., 2020. Is Developer Sentiment Related to Software Bugs: An Exploratory Study on GitHub Commits, in: Int. Conf. Software Analysis, Evolution, and Reengineering (SANER), IEEE. pp. 527–531.

Imran, M.M., Jain, Y., Chatterjee, P., Damevski, K., 2022. Data Augmentation for Improving Emotion Recognition in Software Engineering Communication, in: Proc. Int. Conf. Automated Software Engineering (ASE), ACM. p. 29.

Imtiaz, N., Middleton, J., Chakraborty, J., Robson, N., Bai, G., Murphy-Hill, E., 2019. Investigating the Effects of Gender Bias on GitHub, in: Proc. Int. Conf. Software Engineering (ICSE), IEEE. pp. 700–711.

Imtiaz, N., Middleton, J., Girouard, P., Murphy-Hill, E., 2018. Sentiment and Politeness Analysis Tools on Developer Discussions Are Unreliable, but So Are People, in: Proc. Int. Workshop on Emotion Awareness in Software Engineering (SEmotion), ACM. pp. 55–61.

Infante, D.A., Wigley, C.J., 1986. Verbal Aggressiveness: An Interpersonal Model and Measure. Communication Monographs 53, 61–69.

Islam, M.R., Ahmmed, M.K., Zibran, M.F., 2019. MarValous: Machine Learning Based Detection of Emotions in the Valence-Arousal Space in Software Engineering Text, in: Proc. Sympos. Applied Computing (SAC), ACM. pp. 1786–1793.

Islam, M.R., Zibran, M.F., 2016. Towards Understanding and Exploiting Developers' Emotional Variations in Software Engineering, in: Proc. Int. Conf. Software Engineering Research, Management and Applications (SERA), IEEE. pp. 185–192.

Islam, M.R., Zibran, M.F., 2017a. A Comparison of Dictionary Building Methods for Sentiment Analysis in Software Engineering Text, in: Proc. Int. Sympos. Empirical Software Engineering and Measurement (ESEM), IEEE. pp. 478–479.

Islam, M.R., Zibran, M.F., 2017b. Leveraging Automated Sentiment Analysis in Software Engineering, in: Proc. Int. Workshop on

Mining Software Repositories (MSR), IEEE. pp. 203–214.
Islam, M.R., Zibran, M.F., 2018a. A Comparison of Software Engineering Domain Specific Sentiment Analysis Tools, in: Int. Conf. Software Analysis, Evolution, and Reengineering (SANER), IEEE. pp. 487–491.
Islam, M.R., Zibran, M.F., 2018b. DEVA: Sensing Emotions in the Valence Arousal Space in Software Engineering Text, in: Proc. Sympos. Applied Computing (SAC), ACM. pp. 1536–1543.
Islam, M.R., Zibran, M.F., 2018c. Sentiment Analysis of Software Bug Related Commit Messages, in: Proc. Int. Conf. Software Engineering and Data Engineering (SEDE), International Society for Computers and their Applications (ISCA). p. 740.
Islam, M.R., Zibran, M.F., 2018d. SentiStrength-SE: Exploiting Domain Specificity for Improved Sentiment Analysis in Software Engineering Text. Journal of Systems and Software (JSS) 145, 125–146.
Jalali, S., Wohlin, C., 2012. Systematic Literature Studies: Database Searches vs. Backward Snowballing, in: Proc. Int. Sympos. Empirical Software Engineering and Measurement (ESEM), ACM. pp. 29–38.
Jamieson, J., Yamashita, N., Foong, E., 2024. Predicting Open Source Contributor Turnover from Value-Related Discussions: An Analysis of GitHub Issues, in: Proc. Int. Conf. Software Engineering (ICSE), ACM. pp. 57:1–57:13.
Jongeling, R., Datta, S., Serebrenik, A., 2015. Choosing Your Weapons: On Sentiment Analysis Tools for Software Engineering Research, in: Proc. Int. Conf. Software Maintenance and Evolution (ICSME), IEEE. pp. 531–535.
Jongeling, R., Sarkar, P., Datta, S., Serebrenik, A., 2017. On Negative Results When Using Sentiment Analysis Tools for Software Engineering Research. Empirical Software Engineering (EMSE) 22, 2543–2584.
Jurado, F., Rodriguez, P., 2015. Sentiment Analysis in Monitoring Software Development Processes: An Exploratory Case Study on GitHub's Project Issues. Journal of Systems and Software (JSS) 104, 82–89.
Kadhar, A.A., Kumar, S.S., 2022. Deep-Learning Approach for Sentiment Analysis in Software Engineering Domain, in: Proc. Int. Conf. Information Technology and Applications (ICITA), Springer. pp. 321–330.
Kaur, A., Singh, A.P., Dhillon, G.S., Bisht, D., 2018. Emotion Mining and Sentiment Analysis in Software Engineering Domain, in: Proc. Int. Conf. Electronics, Communication and Aerospace Technology (ICECA), IEEE. pp. 1170–1173.
Kaur, R., Chahal, K.K., Saini, M., 2022. Analysis of Factors Influencing Developers' Sentiments in Commit Logs: Insights from Applying Sentiment Analysis. e-Informatica Software Engineering Journal (EISEJ) 16, 220102.
Kenyon-Dean, K., Ahmed, E., Fujimoto, S., Georges-Filteau, J., Glasz, C., Kaur, B., Lalande, A., Bhanderi, S., Belfer, R., Kanagasabai, N., Sarrazingendron, R., Verma, R., Ruths, D., 2018. Sentiment Analysis: It's Complicated!, in: Proc. Conf. North American Chapter of the Association for Computational Linguistics: Human Language Technologies, ACL. pp. 1886–1895.
Klünder, J., Horstmann, J., Karras, O., 2020. Identifying the Mood of a Software Development Team by Analyzing Text-Based Communication in Chats with Machine Learning, in: Proc. Int. Working Conf. Human-Centered Software Engineering (HCSE), Springer. pp. 133–151.
Kocoń, J., Figas, A., Gruza, M., Puchalska, D., Kajdanowicz, T., Kazienko, P., 2021. Offensive, Aggressive, and Hate Speech Analysis: From Data-Centric to Human-Centered Approach. Information Processing & Management 58, 102643.
Koo, T.K., Li, M.Y., 2016. A Guideline of Selecting and Reporting Intraclass Correlation Coefficients for Reliability Research. Journal of Chiropractic Medicine (JCM) 15, 155–163.
Kraut, R.E., Streeter, L.A., 1995. Coordination in Software Development. Communications of the ACM 38, 69–82.
Krippendorff, K., 2009. Testing the Reliability of Content Analysis Data: What is Involved and Why, in: Krippendorff, K., Bock, M.A. (Eds.), The Content Analysis Reader. SAGE Publications, Inc., pp. 350–357.
Krippendorff, K., 2019. Content Analysis: An Introduction to its Methodology. 4 ed., SAGE Publications, Inc.
Kritikos, A., Venetis, T., Stamelos, I., 2020. An Empirical Investigation of Sentiment Analysis of the Bug Tracking Process in Libre Office Open Source Software, in: Int. Conf. Open Source Systems (OSS), Springer. pp. 36–46.
Kumar, A., Khare, M., Tiwari, S., 2022. Sentiment Analysis of Developers' Comments on GitHub Repository: A Study, in: Int. Conf. Advanced Computational Intelligence (ICACI), IEEE. pp. 91–98.
Kuutila, M., Mäntylä, M.V., Claes, M., 2020. Chat Activity is a Better Predictor than Chat Sentiment on Software Developers Productivity, in: Proc. Int. Conf. Software Engineering Workshops (ICSEW), ACM. pp. 553–556.
Lanovaz, M.J., Adams, B., 2019. Comparing the Communication Tone and Responses of Users and Developers in Two R Mailing Lists: Measuring Positive and Negative Emails. IEEE Software 36, 46–50.
Li, L., Cao, J., Lo, D., 2020. Sentiment Analysis over Collaborative Relationships in Open Source Software Projects, in: Proc. Int. Conf. Software Engineering & Knowledge Engineering (SEKE), KSI Research Inc.. pp. 418–423.
Li, L., Cao, J., Qi, Q., 2021. Monitoring Negative Sentiment-Related Events in Open Source Software Projects, in: Proc. Asia-Pacific Software Engineering Conf. (APSEC), IEEE. pp. 92–100.
Licorish, S.A., MacDonell, S.G., 2014. Relating IS Developers' Attitudes to Engagement, in: Proc. Australasian Conf. Informations Systems (ACIS), ACIS. p. 26.
Licorish, S.A., MacDonell, S.G., 2018. Exploring the Links Between Software Development Task Type, Team Attitudes and Task Completion Performance: Insights from the Jazz Repository. Information and Software Technology (IST) 97, 10–25.
Lin, B., Cassee, N., Serebrenik, A., Bavota, G., Novielli, N., Lanza, M., 2022. Opinion Mining for Software Development: A Systematic Literature Review. ACM Transactions on Software Engineering and Methodology (TOSEM) 31, 38.
Lin, B., Zampetti, F., Bavota, G., Di Penta, M., Lanza, M., 2019. Pattern-Based Mining of Opinions in Q&A Websites, in: Proc. Int. Conf. Software Engineering (ICSE), IEEE. pp. 548–559.
Lin, B., Zampetti, F., Bavota, G., Di Penta, M., Lanza, M., Oliveto, R., 2018. Sentiment Analysis for Software Engineering: How Far Can We Go?, in: Proc. Int. Conf. Software Engineering (ICSE), ACM. pp. 94–104.
Liu, B., 2010. Sentiment Analysis and Subjectivity. Handbook of Natural Language Processing 2, 627–666.
Liu, B., 2017. Many Facets of Sentiment Analysis, in: Cambria, E., Das, D., Bandyopadhyay, S., Feraco, A. (Eds.), A Practical Guide to Sentiment Analysis. Springer.
Liu, B., 2020. Sentiment Analysis: Mining Opinions, Sentiments, and Emotions. Studies in Natural Language Processing. 2 ed., Cambridge University Press.
Madampe, K., Hoda, R., Singh, P., 2020. Towards Understanding Emotional Response to Requirements Changes in Agile Teams, in: Proc. Int. Conf. Software Engineering: New Ideas and Emerging Results (ICSE-NIER), ACM. pp. 37–40.
Mahbub, M., Manjur, N., Alam, M., Vassileva, J., 2021. Analysis of Factors Influencing User Contribution and Predicting Involvement of Users on Stack Overflow, in: Proc. Int. Conf. Educational Data Mining (EDM), International Educational Data Mining Society (IEDMS). pp. 827–832.
Maipradit, R., Hata, H., Matsumoto, K., 2019. Sentiment Classification Using N-Gram Inverse Document Frequency and Automated Machine Learning. IEEE Software 36, 65–70.

Mansoor, N., Peterson, C.S., Sharif, B., 2021. How Developers and Tools Categorize Sentiment in Stack Overflow Questions – A Pilot Study, in: Proc. Int. Workshop on Emotion Awareness in Software Engineering (SEmotion), IEEE. pp. 19–22.

Mäntylä, M.V., Adams, B., Destefanis, G., Graziotin, D., Ortu, M., 2016. Mining Valence, Arousal, and Dominance: Possibilities for Detecting Burnout and Productivity?, in: Proc. Int. Workshop on Mining Software Repositories (MSR), ACM. pp. 247–258.

Mäntylä, M.V., Calefato, F., Claes, M., 2018. Natural Language or Not (NLON): A Package for Software Engineering Text Analysis Pipeline, in: Proc. Int. Workshop on Mining Software Repositories (MSR), ACM. pp. 387–391.

Mäntylä, M.V., Novielli, N., Lanubile, F., Claes, M., Kuutila, M., 2017. Bootstrapping a Lexicon for Emotional Arousal in Software Engineering, in: Proc. Int. Workshop on Mining Software Repositories (MSR), IEEE. pp. 198–202.

Marshall, A., Gamble, R.F., Hale, M.L., 2016. Outcomes of Emotional Content from Agile Team Forum Posts, in: Proc. Int. Workshop on Emotion Awareness in Software Engineering (SEmotion), ACM. pp. 6–11.

Mauerer, W., Joblin, M., Tamburri, D.A., Paradis, C., Kazman, R., Apel, S., 2022. In Search of Socio-Technical Congruence: A Large-Scale Longitudinal Study. IEEE Transactions on Software Engineering (TSE) 8, 3159–3184.

Miller, C., Cohen, S., Klug, D., Vasilescu, B., Kästner, C., 2022. "Did You Miss My Comment or What?" Understanding Toxicity in Open Source Discussions, in: Proc. Int. Conf. Software Engineering (ICSE), ACM. pp. 710–722.

Mohammad, S.M., Zhu, X., Kiritchenko, S., Martin, J., 2015. Sentiment, Emotion, Purpose, and Style in Electoral Tweets. Information Processing & Management 51, 480–499.

Morales-Ramirez, I., Kifetew, F.M., Perini, A., 2019. Speech-Acts Based Analysis for Requirements Discovery from Online Discussions. Information Systems 86, 94–112.

von der Mosel, J., Trautsch, A., Herbold, S., 2023. On the Validity of Pre-trained Transformers for Natural Language Processing in the Software Engineering Domain. IEEE Transactions on Software Engineering (TSE) 49, 1487–1507.

Mostafa, L., Abd Elghany, M., 2018. Investigating Game Developers' Guilt Emotions Using Sentiment Analysis. Int. J. Software Engineering & Application (IJSEA) 9, 16.

Mula, V.K.C., Vijayvargiya, S., Kumar, L., Samant, S.S., Murthy, L.B., 2022. Software Sentiment Analysis Using Machine Learning with Different Word-Embedding, in: Int. Conf. Computational Science and Its Applications Workshops (ICCSAW), Springer. pp. 396–410.

Munaiah, N., Meyers, B.S., Alm, C.O., Meneely, A., Murukannaiah, P.K., Prud'hommeaux, E., Wolff, J., Yu, Y., 2017. Natural Language Insights from Code Reviews that Missed a Vulnerability, in: Int. Sympos. Engineering Secure Software and Systems (ESSoS), Springer. pp. 70–86.

Munezero, M., Montero, C.S., Sutinen, E., Pajunen, J., 2014. Are They Different? Affect, Feeling, Emotion, Sentiment, and Opinion Detection in Text. IEEE Transactions on Affective Computing 5, 101–111.

Murgia, A., Ortu, M., Tourani, P., Adams, B., Demeyer, S., 2018. An Exploratory Qualitative and Quantitative Analysis of Emotions in Issue Report Comments of Open Source Systems. Empirical Software Engineering (EMSE) 23, 521–564.

Murgia, A., Tourani, P., Adams, B., Ortu, M., 2014. Do Developers Feel Emotions? An Exploratory Analysis of Emotions in Software Artifacts, in: Proc. Int. Workshop on Mining Software Repositories (MSR), ACM. pp. 262–271.

Neupane, K.P., Cheung, K., Wang, Y., 2019. EmoD: An End-to-End Approach for Investigating Emotion Dynamics in Software Development, in: Proc. Int. Conf. Software Maintenance and Evolution (ICSME), IEEE. pp. 252–256.

Novielli, N., Calefato, F., Dongiovanni, D., Girardi, D., Lanubile, F., 2020. Can We Use SE-Specific Sentiment Analysis Tools in a Cross-Platform Setting?, in: Proc. Int. Workshop on Mining Software Repositories (MSR), ACM. pp. 158–168.

Novielli, N., Calefato, F., Lanubile, F., 2014. Towards Discovering the Role of Emotions in Stack Overflow, in: Proc. Int. Workshop on Social Software Engineering (SSE), ACM. pp. 33–36.

Novielli, N., Calefato, F., Lanubile, F., 2015. The Challenges of Sentiment Detection in the Social Programmer Ecosystem, in: Proc. Int. Workshop on Social Software Engineering (SSE), ACM. pp. 33–40.

Novielli, N., Calefato, F., Lanubile, F., 2018a. A Gold Standard for Emotion Annotation in Stack Overflow, in: Proc. Int. Workshop on Mining Software Repositories (MSR), ACM. pp. 14–17.

Novielli, N., Calefato, F., Lanubile, F., Serebrenik, A., 2021. Assessment of Off-the-Shelf SE-Specific Sentiment Analysis Tools: An Extended Replication Study. Empirical Software Engineering (EMSE) 26, 77.

Novielli, N., Girardi, D., Lanubile, F., 2018b. A Benchmark Study on Sentiment Analysis for Software Engineering Research, in: Proc. Int. Workshop on Mining Software Repositories (MSR), IEEE. pp. 364–375.

Novielli, N., Serebrenik, A., 2019. Sentiment and Emotion in Software Engineering. IEEE Software 36, 6–23.

Obaidi, M., Holm, H., Schneider, K., Klünder, J., 2022a. On the Limitations of Combining Sentiment Analysis Tools in a Cross-Platform Setting, in: Int. Conf. Product-Focused Software Process Improvement (PROFES), Springer. pp. 108–123.

Obaidi, M., Klünder, J., 2021. Development and Application of Sentiment Analysis Tools in Software Engineering: A Systematic Literature Review, in: Evaluation and Assessment in Software Engineering (EASE), ACM. pp. 80–89.

Obaidi, M., Nagel, L., Specht, A., Klünder, J., 2022b. Sentiment Analysis Tools in Software Engineering: A Systematic Mapping Study. Information and Software Technology (IST) 151, 107018.

Ortu, M., Adams, B., Destefanis, G., Tourani, P., Marchesi, M., Tonelli, R., 2015a. Are Bullies More Productive? Empirical Study of Affectiveness vs. Issue Fixing Time, in: Proc. Int. Workshop on Mining Software Repositories (MSR), IEEE. pp. 303–313.

Ortu, M., Destefanis, G., Counsell, S., Swift, S., Tonelli, R., Marchesi, M., 2016a. Arsonists or Firefighters? Affectiveness in Agile Software Development, in: Proc. Int. Conf. Agile Processes in Software Engineering and Extreme Programming (XP), Springer. pp. 144–155.

Ortu, M., Destefanis, G., Kassab, M., Counsell, S., Marchesi, M., Tonelli, R., 2015b. Would You Mind Fixing this Issue? An Empirical Analysis of Politeness and Attractiveness in Software Developed Using Agile Boards, in: Proc. Int. Conf. Agile Processes in Software Engineering and Extreme Programming (XP), Springer. pp. 129–140.

Ortu, M., Hall, T., Marchesi, M., Tonelli, R., Bowes, D., Destefanis, G., 2018. Mining Communication Patterns in Software Development: A GitHub Analysis, in: Proc. Int. Conf. Predicitive Models in Software Engineering (PROMISE), ACM. pp. 70–79.

Ortu, M., Marchesi, M., Tonelli, R., 2019. Empirical Analysis of Affect of Merged Issues on GitHub, in: Proc. Int. Workshop on Emotion Awareness in Software Engineering (SEmotion), IEEE. pp. 46–48.

Ortu, M., Murgia, A., Destefanis, G., Tourani, P., Tonelli, R., Marchesi, M., Adams, B., 2016b. The Emotional Side of Software Developers in JIRA, in: Proc. Int. Workshop on Mining Software Repositories (MSR), ACM. pp. 480–483.

Pang, B., Lee, L., 2005. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales, in: Proc. Annual Meeting of the Association for Computational Linguistics (ACL), ACL. pp. 115–124.

Panichella, S., Di Sorbo, A., Guzman, E., Visaggio, C.A., Canfora, G., Gall, H.C., 2015. How Can I Improve My App? Classifying User

Reviews for Software Maintenance and Evolution, in: Proc. Int. Conf. Software Maintenance and Evolution (ICSME), IEEE. pp. 281–290.

Park, K.i., Sharif, B., 2021. Assessing Perceived Sentiment in Pull Requests with Emoji: Evidence from Tools and Developer Eye Movements, in: Proc. Int. Workshop on Emotion Awareness in Software Engineering (SEmotion), IEEE. pp. 1–6.

Patnaik, A., Padhy, N., 2022. Sentiment Analysis of Software Project Code Commits, in: Proc. Int. Conf. Next Generation of Internet of Things (ICNGIoT), Springer. pp. 79–88.

Patwardhan, A., 2017. Sentiment Identification for Collaborative, Geographically Dispersed, Cross-Functional Software Development Teams, in: Proc. Int. Conf. Collaboration and Internet Computing (CIC), IEEE. pp. 20–26.

Paul, R., Bosu, A., Sultana, K.Z., 2019. Expressions of Sentiments During Code Reviews: Male vs. Female, in: Int. Conf. Software Analysis, Evolution, and Reengineering (SANER), IEEE. pp. 26–37.

Pavlopoulos, J., Sorensen, J., Dixon, L., Thain, N., Androutsopoulos, I., 2020. Toxicity Detection: Does Context Really Matter?, in: Proc. Annual Meeting Association for Computational Linguistics (ACL), ACL. pp. 4296–4305.

Pletea, D., Vasilescu, B., Serebrenik, A., 2014. Security and Emotion: Sentiment Analysis of Security Discussions on GitHub, in: Proc. Int. Workshop on Mining Software Repositories (MSR), ACM. pp. 348–351.

Prenner, J.A., Robbes, R., 2022. Making the Most of Small Software Engineering Datasets With Modern Machine Learning. IEEE Transactions on Software Engineering (TSE) 48, 5050–5067.

Qiu, H.S., Vasilescu, B., Kästner, C., Egelman, C.D., Jaspan, C., Murphy-Hill, E., 2022. Detecting Interpersonal Conflict in Issues and Code Review: Cross Pollinating Open- and Closed-Source Approaches, in: Proc. Int. Conf. Software Engineering: Software Engineering in Society (ICSE-SEIS), ACM. pp. 41–55.

Quintanilla Portugal, R.L., Sampaio do Prado Leite, J.C., 2018. Usability Related Qualities Through Sentiment Analysis, in: Proc. Int. Workshop on Affective Computing for Requirements Engineering (AffectRE), IEEE. pp. 20–26.

Rahman, M.M., Roy, C.K., Keivanloo, I., 2015. Recommending Insightful Comments for Source Code Using Crowdsourced Knowledge, in: Int. Working Conf. Source Code Analysis and Manipulation (SCAM), IEEE. pp. 81–90.

Raman, N., Cao, M., Tsvetkov, Y., Kästner, C., Vasilescu, B., 2020. Stress and Burnout in Open Source: Toward Finding, Understanding, and Mitigating Unhealthy Interactions, in: Proc. Int. Conf. Software Engineering: New Ideas and Emerging Results (ICSE-NIER), ACM. pp. 57–60.

Ramay, W.Y., Umer, Q., Yin, X.C., Zhu, C., Illahi, I., 2019. Deep Neural Network-Based Severity Prediction of Bug Reports. IEEE Access 7, 46846–46857.

Ramsauer, R., Lohmann, D., Mauerer, W., 2019. The List is the Process: Reliable Pre-Integration Tracking of Commits on Mailing Lists, in: Proc. Int. Conf. Software Engineering (ICSE), IEEE. pp. 807–818.

Rancer, A.S., Avtgis, T.A., 2006. Argumentative and Aggressive Communication: Theory, Research, and Application. SAGE Publications, Inc.

Robbes, R., Janes, A., 2019. Leveraging Small Software Engineering Data Sets with Pre-Trained Neural Networks, in: Proc. Int. Conf. Software Engineering: New Ideas and Emerging Results (ICSE-NIER), IEEE. pp. 29–32.

Robe, P., Kuttal, S.K., AuBuchon, J., Hart, J., 2022. Pair Programming Conversations with Agents vs. Developers: Challenges and Opportunities for SE Community, in: Proc. Europ. Software Engineering Conf. and the Int. Sympos. Foundations of Software Engineering (ESEC/FSE), ACM. pp. 319–331.

Robinson, W.N., Deng, T., Qi, Z., 2016. Developer Behavior and Sentiment from Data Mining Open Source Repositories, in: Proc. Hawaii Int. Conf. System Sciences (HICSS), IEEE. pp. 3729–3738.

Rong, S., Wang, W., Mannan, U.A., de Almeida, E.S., Zhou, S., Ahmed, I., 2022. An Empirical Study of Emoji Use in Software Development Communication. Information and Software Technology (IST) 148, 106912.

Ross, B., Rist, M., Carbonell, G., Cabrera, B., Kurowsky, N., Wojatzki, M., 2016. Measuring the Reliability of Hate Speech Annotations: The Case of the European Refugee Crisis, in: Workshop on Natural Language Processing for Computer-Mediated Communication (NLP4CMC), Ruhr-Universität Bochum. pp. 6–9.

Rousinopoulos, A.I., Robles, G., González-Barahona, J.M., 2014. Sentiment Analysis of Free/Open Source Developers: Preliminary Findings from a Case Study. Revista Eletrônica de Sistemas de Informação (RESI) 13.

Salminen, J.O., Al-Merekhi, H.A., Dey, P., Jansen, B.J., 2018. Inter-Rater Agreement for Social Computing Studies, in: Proc. Int. Conf. Social Networks Analysis, Management and Security (SNAMS), IEEE. pp. 80–87.

Sanei, A., Cheng, J., Adams, B., 2021. The Impacts of Sentiments and Tones in Community-Generated Issue Discussions, in: Proc. Int. Workshop Cooperative and Human Aspects of Software Engineering (CHASE), IEEE. pp. 1–10.

Sapkota, H., Murukannaiah, P.K., Wang, Y., 2019. A Network-Centric Approach for Estimating Trust Between Open Source Software Developers. PLOS ONE 14, e0226281.

Sarker, F., Vasilescu, B., Blincoe, K., Filkov, V., 2019. Socio-Technical Work-Rate Increase Associates With Changes in Work Patterns in Online Projects, in: Proc. Int. Conf. Software Engineering (ICSE), IEEE. pp. 936–947.

Sarker, J., Sultana, S., Wilson, S.R., Bosu, A., 2023a. ToxiSpanSE: An Explainable Toxicity Detection in Code Review Comments, in: Proc. Int. Sympos. Empirical Software Engineering and Measurement (ESEM), IEEE. pp. 283–294.

Sarker, J., Turzo, A.K., Bosu, A., 2020. A Benchmark Study of the Contemporary Toxicity Detectors on Software Engineering Interactions, in: Proc. Asia-Pacific Software Engineering Conf. (APSEC), IEEE. pp. 218–227.

Sarker, J., Turzo, A.K., Dong, M., Bosu, A., 2023b. Automated Identification of Toxic Code Reviews Using ToxiCR. ACM Transactions on Software Engineering and Methodology (TOSEM) 32, 118.

Sayago-Heredia, J., Chango, G., Pérez-Castillo, R., Piattini, M., 2022a. Exploring the Impact of Toxic Comments in Code Quality, in: Proc. Int. Conf. Evaluation of Novel Approaches to Software Engineering (ENASE), SCITEPRESS – Science and Technology Publications. pp. 335–343.

Sayago-Heredia, J., Chango Sailema, G., Pérez-Castillo, R., Piattini, M., 2022b. A Dataset for Analysis of Quality Code and Toxic Comments, in: Int. Conf. Applied Technologies (ICAT), Springer. pp. 559–574.

Scarpa, A., Raine, A., 1997. Psychophyisology of Anger and Violent Behavior. Psychiatric Clinics of North America 20, 375–394.

Schneider, D., Spurlock, S., Squire, M., 2016. Differentiating Communication Styles of Leaders on the Linux Kernel Mailing List, in: Proc. Int. Sympos. Open Collaboration (OpenSym), ACM. p. 2.

Schroth, L., Obaidi, M., Specht, A., Klünder, J., 2022. On the Potentials of Realtime Sentiment Analysis on Text-Based Communication in Software Projects, in: Proc. Int. Working Conf. Human-Centered Software Engineering (HCSE), Springer. pp. 90–109.

Sengupta, S., Haythornthwaite, C., 2020. Learning with Comments: An Analysis of Comments and Community on Stack Overflow, in: Proc. Hawaii Int. Conf. System Sciences (HICSS), University of Hawaii at Manoa. pp. 2898–2907.

Serva, R., Senzer, Z.R., Pollock, L., Vijay-Shanker, K., 2015. Automatically Mining Negative Code Examples from Software Developer Q & A Forums, in: Proc. Int. Conf. Automated Software Engineering Workshops (ASEW), IEEE. pp. 115–122.

Shen, J., Baysal, O., Shafiq, M.O., 2019. Evaluating the Performance of Machine Learning Sentiment Analysis Algorithms in Software

Engineering, in: Int. Conf. Dependable, Autonomic and Secure Computing, Int. Conf. Pervasive Intelligence and Computing, Int. Conf. Cloud and Big Data Computing, Int. Conf. Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), IEEE. pp. 1023–1030.

Singh, N., Singh, P., 2017. How Do Code Refactoring Activities Impact Software Developers' Sentiments? – An Empirical Investigation into GitHub Commits, in: Proc. Asia-Pacific Software Engineering Conf. (APSEC), IEEE. pp. 648–653.

Sinha, V., Lazar, A., Sharif, B., 2016. Analyzing Developer Sentiment in Commit Logs, in: Proc. Int. Workshop on Mining Software Repositories (MSR), ACM. pp. 520–523.

Skriptsova, E., Voronova, E., Danilova, E., Bakhitova, A., 2019. Analysis of Newcomers Activity in Communicative Posts on GitHub, in: Int. Conf. Digital Transformation and Global Society (DTGS), Springer. pp. 452–460.

Sokolovsky, A., Gross, T., Bacardit, J., 2021. Is It Feasible to Detect FLOSS Version Release Events from Textual Messages? A Case Study on Stack Overflow. PLOS ONE 16, e0246464.

Souza, R., Silva, B., 2017. Sentiment Analysis of Travis CI Builds, in: Proc. Int. Workshop on Mining Software Repositories (MSR), IEEE. pp. 459–462.

Stangor, C., Jhangiani, R., Tarry, H., 2014. Principles of Social Psychology. 1 ed., BCcampus.

Steinmacher, I., Treude, C., Gerosa, M.A., 2019. Let Me In: Guidelines for the Successful Onboarding of Newcomers to Open Source Projects. IEEE Software 36, 41–49.

Steinmacher, I., Wiese, I., Chaves, A.P., Gerosa, M.A., 2013. Why Do Newcomers Abandon Open Source Software Projects?, in: Proc. Int. Workshop Cooperative and Human Aspects of Software Engineering (CHASE), IEEE. pp. 25–32.

Storey, M.A., Singer, L., Figueira Filho, F., Zagalsky, A., German, D.M., 2017. How Social and Communication Channels Shape and Challenge a Participatory Culture in Software Development. IEEE Transactions on Software Engineering (TSE) 43, 185–204.

Sun, K., Gao, H., Kuang, H., Ma, X., Rong, G., Shao, D., Zhang, H., 2021. Exploiting the Unique Expression for Improved Sentiment Analysis in Software Engineering Text, in: Proc. Int. Conf. Program Comprehension (ICPC), IEEE. pp. 149–159.

Sun, K., Shi, X., Gao, H., Kuang, H., Ma, X., Rong, G., Shao, D., Zhao, Z., Zhang, H., 2022. Incorporating Pre-trained Transformer Models into TextCNN for Sentiment Analysis on Software Engineering Texts, in: Proc. Asia-Pacific Sympos. Internetware (Internetware), ACM. pp. 127–136.

Swillus, M., Zaidman, A., 2023. Sentiment Overflow in the Testing Stack: Analysing Software Testing Posts on Stack Overflow. Journal of Systems and Software (JSS) 205, 111804.

Tourani, P., Adams, B., 2016. The Impact of Human Discussions on Just-in-Time Quality Assurance: An Empirical Study on OpenStack and Eclipse, in: Int. Conf. Software Analysis, Evolution, and Reengineering (SANER), IEEE. pp. 189–200.

Tourani, P., Jiang, Y., Adams, B., 2014. Monitoring Sentiment in Open Source Mailing Lists: Exploratory Study on the Apache Ecosystem, in: Proc. Int. Conf. Computer Science and Software Engineering (CASCON), IBM Corp.. pp. 34–44.

Uddin, G., Alam, O., Serebrenik, A., 2022a. A Qualitative Study of Developers' Discussions of Their Problems and Joys During the Early COVID-19 Months. Empirical Software Engineering (EMSE) 27, 117.

Uddin, G., Guéhénuc, Y.G., Khomh, F., Roy, C.K., 2022b. An Empirical Study of the Effectiveness of an Ensemble of Stand-Alone Sentiment Detection Tools for Software Engineering Datasets. ACM Transactions on Software Engineering and Methodology (TOSEM) 31, 48.

Uddin, G., Khomh, F., 2021. Automatic Mining of Opinions Expressed About APIs in Stack Overflow. IEEE Transactions on Software Engineering (TSE) 47, 522–559.

Uddin, G., Khomh, F., Roy, C.K., 2020. Mining API Usage Scenarios from Stack Overflow. Information and Software Technology (IST) 122, 106277.

Uddin, G., Khomh, F., Roy, C.K., 2021. Automatic API Usage Scenario Documentation from Technical Q&A Sites. ACM Transactions on Software Engineering and Methodology (TOSEM) 30, 31.

Umer, Q., Liu, H., Illahi, I., 2020. CNN-Based Automatic Prioritization of Bug Reports. IEEE Transactions on Reliability (TRel) 69, 1341–1354.

Valdez, A., Oktaba, H., Gómez, H., Vizcaíno, A., 2020. Sentiment Analysis in Jira Software Repositories, in: Int. Conf. Software Engineering Research and Innovation (CONISOFT), IEEE. pp. 254–259.

Van Atteveldt, W., Van der Velden, M.A., Boukes, M., 2021. The Validity of Sentiment Analysis: Comparing Manual Annotation, Crowd-Coding, Dictionary Approaches, and Machine Learning Algorithms. Communication Methods and Measures 15, 121–140.

Venigalla, A.S.M., Chimalakonda, S., 2021a. StackEmo: Towards Enhancing User Experience by Augmenting Stack Overflow with Emojis, in: Proc. Europ. Software Engineering Conf. and the Int. Sympos. Foundations of Software Engineering (ESEC/FSE), ACM. pp. 1550–1554.

Venigalla, A.S.M., Chimalakonda, S., 2021b. Understanding Emotions of Developer Community Towards Software Documentation, in: Proc. Int. Conf. Software Engineering: Software Engineering in Society (ICSE-SEIS), IEEE. pp. 87–91.

Wang, D., Xiao, T., Son, T., Kula, R.G., Ishio, T., Kamei, Y., Matsumoto, K., 2023. More than React: Investigating the Role of Emoji Reaction in GitHub Pull Requests. Empirical Software Engineering (EMSE) 28, 123.

Wang, Y., 2019. Emotions Extracted from Text vs. True Emotions – An Empirical Evaluation in SE Context, in: Proc. Int. Conf. Automated Software Engineering (ASE), IEEE. pp. 230–242.

Waseem, Z., Davidson, T., Warmsley, D., Weber, I., 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks, in: Proc. Workshop on Abusive Language Online (ALW), ACL. pp. 78–84.

Werder, K., 2018. The Evolution of Emotional Displays in Open Source Software Development Teams: An Individual Growth Curve Analysis, in: Proc. Int. Workshop on Emotion Awareness in Software Engineering (SEmotion), ACM. pp. 1–6.

Werder, K., Brinkkemper, S., 2018. MEME: Toward a Method for Emotions Extraction from GitHub, in: Proc. Int. Workshop on Emotion Awareness in Software Engineering (SEmotion), ACM. pp. 20–24.

Werner, C., Li, Z.S., Damian, D., 2019. Can a Machine Learn Through Customer Sentiment?: A Cost-Aware Approach to Predict Support Ticket Escalations. IEEE Software 36, 38–45.

Werner, C., Tapuc, G., Montgomery, L., Sharma, D., Dodos, S., Damian, D., 2018. How Angry Are Your Customers? Sentiment Analysis of Support Tickets that Escalate, in: Int. Workshop on Affective Computing for Requirements Engineering (AffectRE), IEEE. pp. 1–8.

Wu, J., Ye, C., Zhou, H., 2021. BERT for Sentiment Classification in Software Engineering, in: Int. Conf. Service Science (ICSS), IEEE. pp. 115–121.

Wulczyn, E., Thain, N., Dixon, L., 2017. Ex Machina: Personal Attacks Seen at Scale, in: Proc. Int. Conf. World Wide Web (WWW), ACM. pp. 1391–1399.

Yang, B., Wei, X., Liu, C., 2017a. Sentiments Analysis in GitHub Repositories: An Empirical Study, in: Proc. Asia-Pacific Software Engineering Conf. Workshops (APSECW), IEEE. pp. 84–89.

Yang, G., Baek, S., Lee, J.W., Lee, B., 2017b. Analyzing Emotion Words to Predict Severity of Software Bugs: A Case Study of Open Source Projects, in: Proc. Sympos. Applied Computing (SAC), ACM. pp. 1280–1287.

Yang, G., Zhang, T., Lee, B., 2018. An Emotion Similarity Based Severity Prediction of Software Bugs: A Case Study of Open Source Projects. IEICE Transactions on Information and Systems (Trans. Info. Syst.) 101, 2015–2026.

Yin, L., Zhang, X., Filkov, V., 2023. On the Self-Governance and Episodic Changes in Apache Incubator Projects: An Empirical Study, in: Proc. Int. Conf. Software Engineering (ICSE), IEEE. pp. 678–689.

Zhang, D., Dai, D., Han, R., Zheng, M., 2021. SentiLog: Anomaly Detecting on Parallel File Systems via Log-Based Sentiment Analysis, in: Proc. Workshop Hot Topics in Storage and File Systems (HotStorage), ACM. pp. 86–93.

Zhang, T., Xu, B., Thung, F., Haryono, S.A., Lo, D., Jiang, L., 2020. Sentiment Analysis for Software Engineering: How Far Can Pre-trained Transformer Models Go?, in: Proc. Int. Conf. Software Maintenance and Evolution (ICSME), IEEE. pp. 70–80.

Zhang, Y., Hou, D., 2013. Extracting Problematic API Features from Forum Discussions, in: Proc. Int. Conf. Program Comprehension (ICPC), IEEE. pp. 142–151.

Thomas Bock currently is a post-doctoral researcher at the Chair of Software Engineering at Saarland University & Saarland Informatics Campus, Germany. He received his Ph.D. in Computer Science in 2024 from Saarland University. His research interests are focused on empirical software engineering, software evolution and maintenance, and social aspects of software engineering.

Niklas Schneider is a trainee teacher for Computer Science and Mathematics and studied Computer Science and Teaching at Saarland University & Saarland Informatics Campus, Germany, where he was concerned with socio-technical and educational aspects of software engineering as a student research assistant.

Angelika Schmid is a data scientist and managing consultant. She has been working as a research assistant in the junior research group PICCARD at the University of Passau, Germany, where she has been involved in the analysis of developer networks with respect to social structures, as well as at the Chair of Statistics at the University of Passau. She has received her Ph.D. in Applied Statistics from the University of Passau in 2019. Before, she had studied Intercultural Communication, Business Administration, and Economics at the University of Passau.

Sven Apel is a professor and holds the Chair of Software Engineering at Saarland University & Saarland Informatics Campus, Germany. Prof. Apel received his Ph.D. in Computer Science in 2007 from the University of Magdeburg. His research interests include software product lines, software analysis, optimization, and evolution, as well as empirical methods and the human factor in software engineering.

Janet Siegmund is a professor for Software Engineering. Before, she led the junior research group PICCARD, funded by the Center Digitisation.Bavaria. She received her Ph.D. from the University of Magdeburg, Germany, in 2012 and holds two master's degrees (Computer Science and Psychology). Her research is centered around the human factor in software engineering. She regularly serves as program-committee member or chair for conferences and workshops and was in the steering committee of the International Conference on Program Comprehension.