An Empirical Study of Sentiments in Code Reviews

Ikram El Asri, Noureddine Kerzazi, Gias Uddin, Foutse Khomh, M. A. Janati Idrissi

Mohammed V University in Rabat, Morocco, ENSIAS.
Polytechnique Montreal, Canada.

Abstract

Context: Modern code reviews are supported by tools to enhance developers' interactions allowing contributors to submit their opinions for each committed change in form of comments. Although the comments are aimed at discussing potential technical issues, the text might enclose harmful sentiments that could erode the benefits of suggested changes.
Objective: In this paper, we study empirically the impact of sentiment embodied within developers' comments on the time and outcome of the code review process.
Method: Based on historical data of four long-lived Open Source Software (OSS) projects from a code review system we investigate whether perceived sentiments have any impact on the interval time of code changes acceptance.
Results: We found that (1) contributors frequently express positive and negative sentiments during code review activities; (2) the expressed sentiments differ among the contributors depending on their position within the social network of the reviewers (*e.g.,* core vs peripheral contributors); (3) the sentiments expressed by contributors tend to be neutral as they progress from the status of newcomer in an OSS project to the status of core team contributors; (4) the reviews with negative comments on average took more time to complete than the reviews with positive/neutral comments, and (5) the reviews with controversial comments took significantly longer time in one project.
Conclusion: Through this work, we provide evidences that text-based sentiments have an impact on the duration of the code review process as well as the acceptance or rejection of the suggested changes.

Keywords: Empirical Software Engineering, Code review, Sentiment Analysis, Opinion Mining, Affective Analysis, Propensity Score Matching.

1. Introduction

Peer code review is the practice where a developer submits a piece of code (*i.e.,* code changes) to peers to judge its eligibility to be integrated into the main project code-base [1]. It aims to assess the quality of source code changes made by contributors before they are integrated into the mainstream. Beyond technical information, the textual comments of reviews could contain either positive or negative sentiments, which might alter the perception of their benefits. Past studies have shown that mailing lists of virtual communities include not only useful information such as ideas for improvements, but also contributor opinions, and feelings about the introduced changes [2]. There are also evidences that developers' opinions play a key role in the decision-making process of source code reviews [3, 4, 5]. However, little is known about the impact of the expressed sentiments on the effectiveness of the review process.

Previously, Baysal *et al.* [6] have explored the impact of technical and non-technical factors on the duration of source code reviews. They observed that non-technical factors, such as reviewer experience can significantly impact code review outcomes. An empirical understanding of the impact of sentiments in code review process can add a novel dimension to the findings of Baysal *et al.* [6] – notably to guide the design of better code review approaches and tools to facilitate improved productivity.

With a view to understand the prevalence and impact of sentiments in modern code reviews, we empirically studied the code reviews of four long-lived software projects. In particular, we answer four research questions:

RQ1: What is the performance of Sentiment Detectors When Applied on Code Reviews?

Recent studies [7, 8, 9] have raised uncertainties related to the unsuccessful application of sentiment analysis tools for software engineering. Indeed, existing tools might require customization to satisfy needs of a specific usage context such as technical software engineering. Following Novielli et al. [10], we carried out a benchmark-based study of three sentiment detection tools that are widely used in software engineering research (*Senti4SD* [11], *SentiCR* [12], and *Sentistrength_SE* [13]). We found that Senti4SD tool provides the best performance (F1 79) when applied to our code review samples datasets. We used Senti4SD [11] in our subsequent analysis.

RQ2: How Prevalent are Sentiments in Code Reviews?

We found that contributors express sentiments in their review comments (**13.94% of comments were positive**, **2.24% negative**, and **83.81% were identified as neutral**). We observed that both core and peripheral contributors do express sentiments in the code reviews. Core members are those developers that contribute intensively and consistently to the OSS project, and thus, lead the community, while peripheral ones are occasional contributors with less frequent commits. We built Social Network Graphs of reviewers to segregate Core and Peripheral contributors. Our analysis reveals that the sentiments of Core contributors tend to become more neutral over time.

RQ3: How do the presence of sentiments in code reviews correlate with the outcome of the reviews?

We examined the effect of sentiments on the outcome of code reviews. We observed that reviews with negative comments on average take longer time to complete. In contrast, the reviews with positive sentiments had a lower duration. Reviews that contain positive sentiments required, on average, **1.32 day less time** to be closed than those with negative sentiments. Moreover, we found that **91.81%** of successful reviews were identified with positive sentiments, and **64.44%** of aborted reviews contained negative sentiments.

Contributions. This paper makes the following contributions:

1. We provide empirical evidence on the effect of expressed sentiments on the outcome of code reviews. Providing stakeholders with a better understanding of the impact of contributors' sentiments on team dynamics and their productivity;

2. We investigate whether the core (*i.e.,* experienced) developers and the peripheral (*i.e.,* newcomers) developers express different types of sentiments and the effect of these sentiments on the efficiency of code reviews;

3. We monitor the evolvement of sentiments of the top 5% contributors across time, for four OSS projects, as they progress and gain more experience, aiming at understanding the correlation between notoriety (*i.e.,* experience) and the trend of sentiments expressed in text-based interactions.

Paper organization. Section 2 provides background information on sentiment analysis, the code review process, and the social network analyses conducted in this paper. Section 3 discusses the related literature. Section 4 describes the methodology of our case study. Section 5 reports our findings. Section 6 discusses our results. Section 7 highlights threats to the validity of our study and Section 8 concludes the paper and outlines directions for future work.

2. Background

This section provides background information about sentiment analysis, code review, and social network analysis.

2.1. What Does Sentiment Analysis Stand for?

Emotion and sentiment are terms relating to human subjectivity [14] understood in the same way and used interchangeably in different domains even if they are not synonymous. Sentiment detection focuses on the detection of subjectivity in a given input (*e.g.,* a sentence). A subjectivity can be of three types: (1) Positive, (2) Negative, and (3) Neutral. Emotion detection focuses on a finer-grained detection of the underlying expressions carried over by the sentiments, such as, anger, frustration. Gerrod Parrott identified six prominent emotions in social psychology [15]: *(1)* Joy, *(2)* Love, *(3)* Surprise, *(4)* Anger, *(5)* Sadness, and *(6)* Fear. This paper focuses on the analysis of sentiments in code reviews, because sentiment detection is predominantly used in other domains (*e.g.,* cars, movies) to mine and summarize opinions about entities [16]. Although, analyzing sentiments and emotions in text data similarly related to one another, actually the granularity is quite different. For example, *"this new feature wasn't what I expected"* and *"I hate using this API with buggy source code"* are both negative sentiments. While a Sentiment Analysis seeks to catch the general feel or impression people get from consuming a piece of content, Emotion Analysis stresses the specific articulate emotions such as happy, angry, sad, etc.

Sentiment analysis can be performed typically at one of the three levels: document level, sentence level, feature level [17]. In this study, we perform a document level analysis.

2.2. Modern Code Review Practice

Code change review is a well-established practice to improve code quality in software engineering. Developers read and assess each other's code change before it is integrated into the mainstream line of code towards a release. Gerrit[1] is one of the tools providing infrastructure for online reviews as a substitute to face-to-face meetings or mailing lists. It is an online tool that supports the traceability of the code review process by explicitly linking changes to a software system recorded in a Version Control System (VCS) to their respective code review discussions.

Figure 1 illustrates the overall process underpinning the code review flow into Gerrit tool. There are three roles into Gerrit: Author, Reviewer, and Verifier as shown in Figure 1. Authors commit code changes into VCS and request a review. Reviewers are responsible for passing throughout the changes and then proposing and discussing adjustments within comments. In other words, reviewers might spot potential defects that authors are not consciously aware of. Then, the author addresses the comments and produces a new code revision. Verifiers are responsible for executing tests to ensure that proposed changes are bug-free and do not cause any regression of the system. They can also leave comments to describe verification issues that they might encounter during testing. Once the criteria for a review are satisfied, changes are integrated into the mainstream repository and flagged as "Merged". This lifecycle may have another different transition "Abandoned" when the review has not passed the evaluation and is no longer active.

2.3. Code Review Factors

One of the main concern of developers when submitting patches for code review is maximizing the chances of their patches being examined in the shortest possible time. However, the outcome and duration of the code review process can be affected by a variety of technical factors. These influencing factors might introduce some bias when analyzing the real effect of contributor's sentiments on review fixing time and review outcome.

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

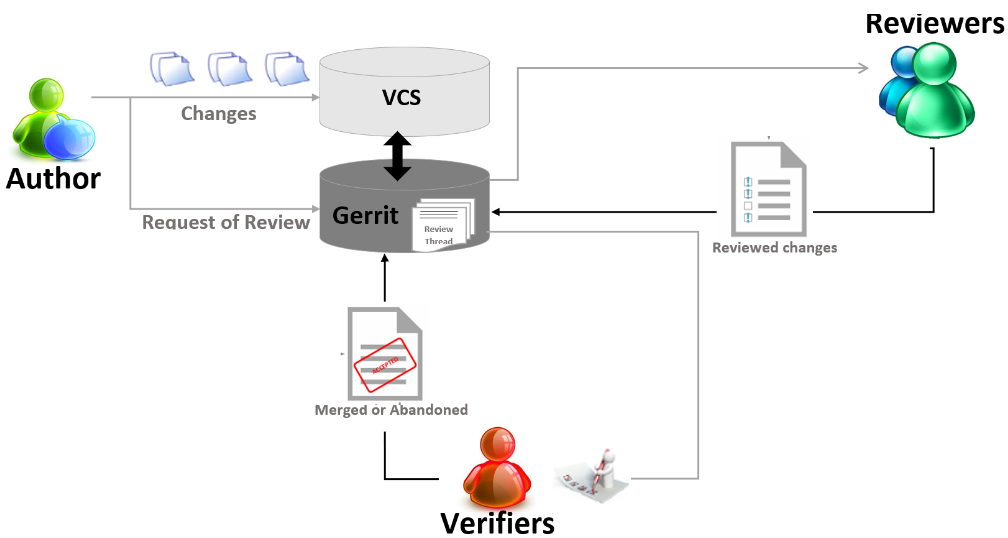

Figure 1: Code Review Flow

The most intuitive factor is patch size (Churn); previous studies have found that smaller patches are more likely to receive faster responses [18, 19] since larger patches would be more difficult to review, and hence require more time. Another important factor is how many times a developer had to resubmit his patch for an additional review (Count Patches); a patch requiring multiple revisions and re-submission(s) before being accepted consumes more time. Moreover, the more wide-spread a change is across files (Edited Files), the more concepts it touches in a system which often results in more rework [20]. Based on a survey of 88 open source core developers, Kononenko et al. [21] confirmed the important influence of these technical factors on reviews process and outcome. Authors report that the length of the discussion (count comments) and the amount of people involved in the discussion (Distinct involved Contributors) were judged as influencing factors by the interviewed contributors.

For our study, we select these widely used technical metrics to characterize reviewed patches. Then, we use the propensity score matching (PSM) technique (see Section 4.3) to ensure that our analysis is not biased by different technical characteristics. PSM [22] is a statistical matching technique that allows us to create groups of reviews that share similar characteristics. The technical characteristics considered in our study are:

- Count Comments : The number of comments posted on each code review request (*i.e.,* about the proposed code change).

- Count Patches : The number of patches submitted before the proposed code change is accepted or rejected.

- Edited Files (discrete count) : The number of files modified by the proposed code change.

- Distinct involved Contributors : The number of developers that participated in the review of the proposed code change.

- Code Churn (Cumulative count) : The number of added and deleted lines that are performed in the reviewed code changes.

2.4. Social Network Analysis

Social Network Analysis (SNA) is the process of investigating social structures through the use of networks and graph theory [23]. A Network is typically modeled using a graph structure consisting of vertices and edges. Vertices represent individuals or organizations. An edge connecting two vertices represents some type of relationships between the two individuals or organizations. Social network analysis focuses on studying social network graphs to understand the patterns of interactions and the relative positions of individuals in a social setting [24]. SNA provides various global or node-specific computed metrics for a network, that are useful for making general statements about specific nodes or classes of nodes. Examples of such metrics are betweenness, diameter, distance, density, betweenness centrality, degree centrality, or eigenvector centrality [23].

SNA is being widely used by researchers to model the social structure of OSS communities and barely used in analyzing Open Source Software Peer Review [25]. Previous studies using SNA in OSS generally indicated a few central persons being responsible for most of the interactions in the network *(Core)* and a less connected large group of contributors *(Peripheral)* [26]. Through sentiment analysis, we aim to get insights about contributor's positivity/negativity in relation to their position in code review interactions networks.

3. Related Works

Several works have focused the attention of the research community on sentiments analysis. These works span many fields ranging from happiness at workplaces [27] to emotions in social networks' messages such as Yahoo and Twitter [4, 28] and online Q&A such as Stack Overflow posts [29]. Guillory *et al.* [30] went a step further and examined the spread of negative emotions into online communities. Their analyses suggest that contagion of negative emotions can occur in groups of people and impact their performance.

Guzman and Bruegge [31] presented a position paper that describes emotional awareness in software development teams. The paper was motivated by the same concerns that have motivated our approach. Their approach investigates the collective emotional awareness of developers in distributed teams. It extracts emotional state from a 1,000 of collaboration artifacts aiming to summarize emotions expressed in those artifacts by extracting topics and assigning them an average emotion score. Authors presented the emotion average fluctuation to the project leaders, whom confirmed the correlation of positive and negative emotion peaks with team performance, motivation and important deadlines. Our work improves and expands their idea by using propensity score to allow for more accurate comparisons, and apply them on comments related to code reviews instead of comments from commits.

Sinha *et al.* [32] analyzed developers commits logs for a large set of Github projects and found that the majority of the sentiment expressed by developers is neutral. They also found that negative comments are more present than positive ones (respectively 18.05% vs. 7.17%). Similarly, Guzman *et al.* [33] examined the sentiments expressed by developers in comments related to commits from 29 open source projects and found an approximately equal distribution of positive, negative and neutral sentiments. Paul et al. [34] explored the difference of expressed sentiments between men and women during various software engineering tasks including the code review practice. The authors report that women are less likely to express their sentiment than men and that sentiment words, emoticons, and expletives vary cross-gender. However, their study did not investigate the effect of expressed sentiment on the prodctivity of the code review activity according to the duration and results.

Khan *et al.* [35] conducted two studies to explore the impact of sentiments on developer's performance. They found that programmers' moods influence positively some programming tasks such as debugging. Similarly, Ortu *et al.* [36] studied the impact of developers' affectiveness on productivity focusing on the correlation between emotional states and productivity in terms of issues fixing time. They report that the happier developers are, *i.e.,* expressing emotions such as joy and love in their comments, the shorter

Table 1: Existing Sentiment Analysis Tools.

Tool	Purpose	Technique	Trained on	Ref.
Sentistrength	General	Rule-based	Twitter	[28]
Sentistrength_SE	Focused	Rule-based	Jira	[13]
Senti4SD	Focused	Lexical Features	Stack Overflow	[11]
SentiCR	Focused	Lexical Features	Code Reviews	[12]

the issue fixing time is likely to be. They also report that emotions such as sadness are linked to longer issue fixing time. Also, Destefanis *et al.* [37] investigated social aspects among developers working on software projects and explored whether the politeness of comments affected the time required to fix any given issue. Their results showed that the level of politeness in the communication process among developers does have an effect on the time required to fix issues and, more specifically the more polite the developers were, the less time it took to fix an issue. We complement existing work on the impact of sentiment on productivity by studying the influence of text-based expressed sentiment on the duration and outcome of code reviews.

Recent studies have investigated factors affecting the effectiveness of code review comments. Rahman *et al.* [38] extracted a number of features from the text of the review comments attempting to predict the usefulness of code review comments using textual features. However, their empirical study was limited to structural characteristics of the text without considering emotions/sentiments expressed in them. Efstathiou and Spinellis [7] studied the language of code review comments and report that language does matters. In this paper, we continue this line of work by investigating the role of sentiments expressed in code review comments on the outcome of code review. Since Lin et al. [8] recently highlighted issues with the accuracy of existing sentiment analysis tools from the literature, we have choose the most powerful sentiment analysis tool based on a benchmarking of several sentiment analysis tools. In Table 1, we present a summary of existing sentiment analysis tools that are designed and tested using data from software artifacts.

4. Empirical Study Design

Our overall goal is to understand the influence of expressed sentiment, throughout comments, on time and outcomes of code reviews. Figure 2

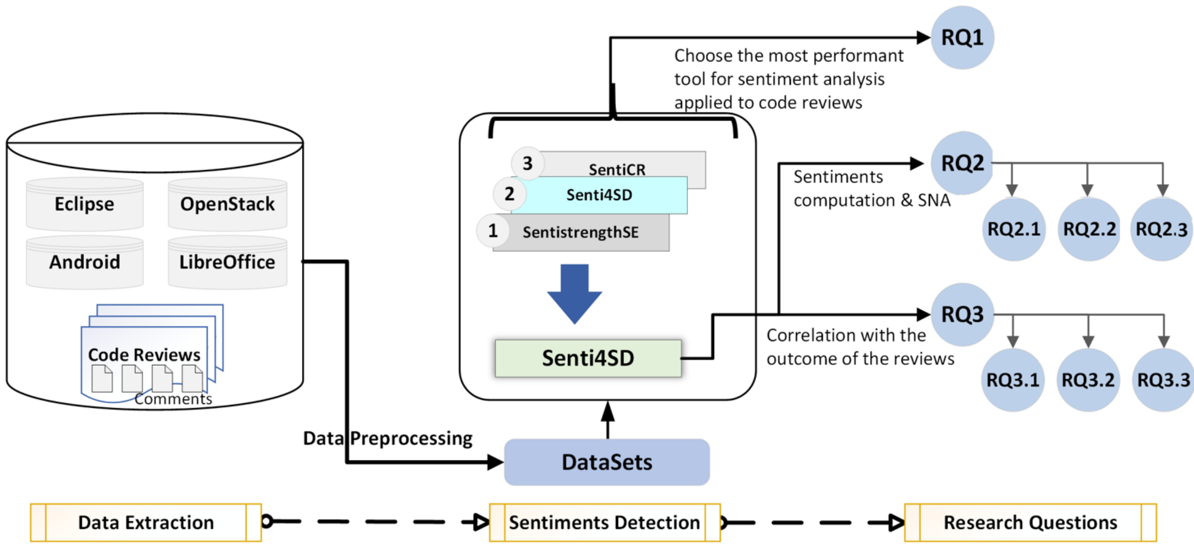

Figure 2: Overview of Our Empirical Study.

presents an overview of the steps of our study and how they relate to our research questions. In the remainder of this section, we describe each step in details.

4.1. Data Collection

We conduct our empirical study based on publicly available code review data, mined from Gerrit system and organized in a portable database dump [39]. We selected this data set because it contains a substantial volume of data from well-known open source projects organized in a relational database[2] as depicted in Figure 3. In our study we used data of four well-known open-source systems, OpenStack[3], Eclipse[4], Android[5] and LibreOffice[6]. OpenStack is a software platform for cloud computing, controlling large pools of computing, storage, and networking resources throughout a data center. Eclipse is an integrated development environment (IDE) used in computer programming. Android is a free software stack for a wide range

[2] http://kin-y.github.io/miningReviewRepo/
[3] http://openstack.org
[4] https://eclipse.org/
[5] https://source.android.com/
[6] https://www.libreoffice.org/

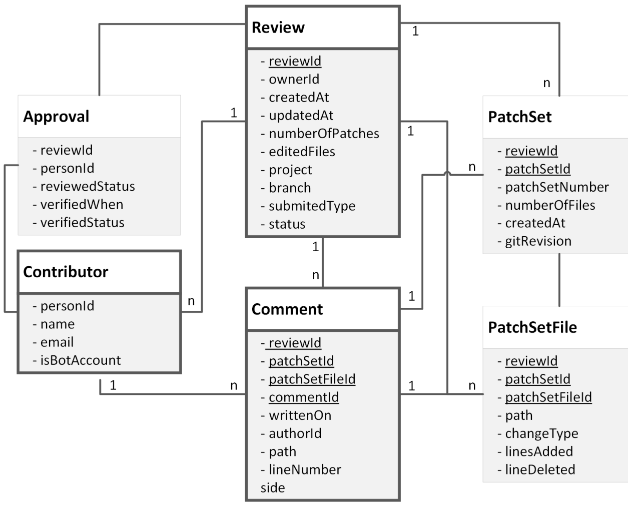

Figure 3: Simplified database schema of Gerrit data.

of mobile devices led by Google. LibreOffice is a fork from the OpenOffice.org project. We selected these projects because they have been actively developed for more than five years and hence provide a rich data set of reviews. Also, they are from different domains, are written in different programming languages, and have been quite studied in other research domains.

The original dataset[7] is stored in a relational database (335,626 reviews and contains over 5 million comments) under the schema depicted in Figure 3. In general, a contributor (*i.e.,* personId, name, email) requests a review characterized by a reviewId, the creation time (createdAt), the last time modified (updatedAt), related project and the source code branch. A review includes a set of patches when the author repeatedly update the change by committing new resubmissions with the same review request ID and a list of edited files. The history of launched discussion over proposed changes is recorded in the table '*Comment*'.

We retrieved and exported required data into separate csv files to ease our data pre-processing. Table 2 shows descriptive statistics regarding the studied projects.

[7]http://kin-y.github.io/miningReviewRepo/

Table 2: A statistical summary for each studied system

Projects	#Reviews	#Comments	#Contributors
Openstack	228,099	5,021,264	8,088
Eclipse	15,887	153,176	1,082
Android	63,610	355,765	3,334
LibreOffice	28,030	174,181	634

4.2. Data Preprocessing

In order to improve the quality of our dataset with respect to our main goal which is studying the human sentiments expressed in code review comments, we performed three pre-processing steps on the raw data:

1. We discarded comments generated automatically such as those generated by build automation and continuous integration services. Those comments contain key-words such as: ‘Jenkins’, ‘Hudson’, ‘Bot’, or ‘CI Servers’ (about 29% of the total comments were excluded). Using regular expressions, we excluded automatic expressions (*e.g.,* Build succeed, Build failed, etc.). In addition, we removed reviews with status = “New” since their final status remains unknown ($\sim$ 5% of total reviews). We limited our analysis to closed reviews (i.e., reviews marked as ”Merged” or ”Abandoned”) that contain at least one comment. The remaining data set contains 4,426,451 comments belonging to 317,373 reviews.

2. For each review, we gathered key information such as timestamp of opening and closing of the review, count of edited files, count of patches, and number of added and deleted lines. Clustering reviews based on these metrics help us to make unbiased comparisons later on.

4.3. Data Analysis

We use the propensity score matching (PSM)[8] [22] method to regroup reviews homogeneously according to some characteristics (*e.g.,* size of the review, code churn, number of comments, etc.). Previous works report that code reviews are affected by a variety of technical factors such as the size

[8]https://en.wikipedia.org/wiki/Propensity_score_matching

of the source code [6]. Using PSM allows us to be able to compare reviews that are logically comparable in terms of these known affecting factors. PSM is a statistical matching technique widely used to compress covariates to a variable (*i.e.,* compare technical factors and generate a propensity score). PSM is proposed to treat the effects of confounding factors. [40] presents an evaluation of the efficiency of PSM in mitigating confounding factors.
In this work, we used the R package called *Matchit* to carry out the first two steps enumerated bellow, while step 3 required a manual verification. The three steps are described as follows:

1. A logistic regression model is built based on a high-dimensional set of characteristics. The revision sentiment (Positive or Negative) is set as the dependent variable, and reviews technical characteristics: *(i)* amount of comments for the review; *(ii)* count of patchSets, *(iii)* number of edited files, *(iv)* distinct involved Contributors, and *(v)* churn are set as its independent variables. The output of the logistic regression model is a fitted value (a probability value) called propensity score.

2. The propensity scores are used to match pairs of data points. Each pair has different values of the dependent variable. Similar values of the propensity score imply a similarity of reviews technical characteristics. For our purpose we used the Genetic matching algorithm to match appropriate pairs of reviews. Then matched pairs are combined into a new dataset.

3. The final step is to verify the balance of covariate characteristics. To do that, we manually compared the means differences for each covariate variable across matched reviews.

The output of PSM is two groups : one for positive reviews and the other one for negative reviews. Although the final step of PSM is to verify the balance of covariate characteristics, we carried out a manual validation of confounding bias on PSM outputs as shown in Table 3. For instance, the mean difference of total comments for positive and negative reviews shift respectively from (9.96, 12.6) to (6.2, 6.2), which means more homogeneous groups. One can also notice greater p-values[9] with matched reviews; meaning

[9] The p-values are calculated using Mann-Whitney U test [41]

an equivalent distribution regarding technical characteristics.

Table 3: Mean differences of technical characteristics before and after Propensity Score Matching (Eclipse Project).

	Befor PSM			After PSM		
	Positive reviews	Negative reviews	p-value	Positive reviews	Negative reviews	p-value
Comments	9.96	12.60	9.07e-06	6.20	6.20	0.09
Patchesets	2.60	2.63	2e-03	1.51	1.51	0.06
Edited_files	13.25	14.22	0.33	1.66	1.66	0.37
Churn	4993.70	8431.20	0.06	51.49	42.29	0.34
Distinct contrib	2.94	3.27	1.60e-7	2.66	2.66	2e-4

The new balanced dataset, used later in answering RQ3.1, contains (2,393 positive vs 876 negative reviews) for Openstack, (811 positive vs 155 negative reviews) for Eclipse, (11,831 positive vs 2,373 negative reviews) for Android, and (9,002 positive vs 498 negative reviews) for LibreOffice.

5. Findings

We now present the findings of the sentiment analysis conducted on four OSS projects. For each of our four research questions, we present our motivation, the approach, and results.

RQ1. What is the performance of Sentiment Detectors When Applied on Code Reviews?

• **Motivation.** To investigate whether the negative and positive sentiments expressed in developers' text-based review interactions affect the code review process, we need a tool capable of detecting sentiments in code review comments accurately. So far three different sentiment detection engines have been proposed in the software engineering literature [13, 11, 12], but not trained specifically on comments of code reviews. However, these previous attempts to analyze text-based sentiments for software engineering have been either incomplete nor reusable for other domains [8, 7]. A major reason of this inadequacy is that software engineering encompasses vocabulary from diverse sub-domains [7]. Therefore, a tool trained and successfully tested in one sub-domain (e.g., Q&R Stack Overflow) may not be useful enough for another sub-domain (i.e., comments within Jira issues system). Consequently, we were more cautious on how to choose our tools.

• **Approach.** We compared the performance of three sentiment detection tools: SentistrengthSE [13], Senti4SD [11], and SentiCR [12] aiming to choose the most adequate tool for the domain of source code reviews. These three tools have been trained previously to detect sentiments in software engineering using specific datasets (see Table 1). In order to compare cautiously the performance of the three tools, we carried out a manual annotation (by four raters) on a subset of comments. To strengthen our sampling, we built an over-sampling approach in which the minority class (*i.e.,* negative sentiments) is equally represented. Concretely, following the approach used by Novieli et al. [42], we built four sub-datasets by performing opportunistic sampling. The first sample is created based on the output of SentiStrength_SE, it contains 1,200 comments equally distributed (for each project we have 100 Positive, 100 Negtive and 100 Neutral, making $300X4 = 1,200$). The second and the third samples retrieved respectively from Senti4SD and SentiCR contain 360 comments each. The final sample contains 300 random comments. Then each review comment was manually annotated (positive, negative, neutral) by the first author and one of the other authors to ensure a stable annotation. The same approach was previously used by Lin et al. [8] to produce their sentiment benchmark. The agreement between the two coders, measured using Cohen's kappa, ranged from 81% to 95% (83% for Senti4SD sample, 95% for SentiCR sample, 81% for SentiStrengthSE and 91% for Random sample).To resolve the disagreements between raters, the annotations were discussed and the guideline of annotation was updated by the first author. For instance, an example of a disagreement between two annotators happened fir the following sentence: "*Patch Set 2: Fails Merges in public tree, but does not build. Please fix and reupload. Thanks!*". The first annotator classified this comment as Positive, while the second one classified it as Negative. Through mutual consent we decided to tag this typical comment as Positive since the commenter was very polite and used "Please" and "Thanks" in his text.

Our sub-datasets as well as the original dataset (5 millions comments) are available in the companion on line appendix [43] for the purpose of replication.

• **Results.** Table 4 reports the performance obtained in terms of recall, precision, and F1-measure, for each polarity classes (Positive, Negative, and Neutral) as well as the overall performance, for the three tools when applied to our data samples. We highlight the best values for each metric. Surprisingly, when expecting a good performance from SentiCR tool, Senti4SD shows a slightly better overall performance than the other tools (F1 = 0.79[10]). Again, Lin et al [8] pointed out that sentiment analysis tools should always be carefully evaluated in the specific context of usage. We double check our results by performing McNemar[11] statistical test [45] in order to compare the classification results of the three tools. The performance differences between Senti4SD and other classifiers were found to be statistically significant ($p\_value \prec 0.05$ and z scores $= 11.49 \succ 0$) indicating that Senti4SD performs better than SentiCR. Moreover, when comparing this result with our manual tagging, we noticed that 472 comments were correctly classified by Senti4SD and misclassified by SentiCR while only 178 comments correctly classified by SentiCR and misclassified by Senti4SD.

RQ1 What is the performance of Sentiment Detectors When Applied on Code Reviews?

On average Senti4SD led to the best performance (Precision **79%**, F1 **79%**) when applied to our code review data samples.

RQ2. How Prevalent are Sentiments in Code Reviews?

• **Motivation.** Sentiments are ubiquitous in human activity: There is an old saying "*Feeling Good-Doing Good*" [46]. OSS contributors may underperform if they do not feel safe and happy [35]. Negative emotions like anger can make people less motivated and thus less creative [36]; two key factors to ensure productivity within modern software organizations [27]. For instance, Linus Torvalds sent out an email[12] to the Linux developers' community admitting his verbal abuse in communications *["My flippant attacks*

[10]We used the F1-measure to determine the best performing classifiers, following standard practices in Information Retrieval [44]

[11]https://stat.ethz.ch/R-manual/R-devel/library/stats/html/mcnemar.test.html

[12]https://gizmodo.com/linux-founder-takes-some-time-off-to-learn-how-to-stop-1829105667

Table 4: Performance of Sentiment Detectors in Code Review samples (P = Precision, R = Recall, F1 = F1-Measure)

Dataset	Class	Sentistrength_SE P	Sentistrength_SE R	Sentistrength_SE F1	Senti4SD P	Senti4SD R	Senti4SD F1	SentiCR P	SentiCR R	SentiCR F1
SentistrengthSE_based	Positive	**0.86**	0.85	**0.83**	0.81	0.84	0.82	0.59	**0.89**	0.71
	Negative	**0.91**	0.61	**0.73**	0.66	**0.68**	0.67	0.59	0.66	0.62
	Neutral	0.7	**0.98**	0.81	0.83	0.81	**0.82**	**0.88**	0.69	0.77
	Micro-avg.	**0.8**	**0.8**	**0.8**	0.79	0.79	0.79	0.72	0.72	0.72
	Macro-avg.	**0.82**	**0.8**	**0.79**	0.77	0.77	0.77	0.69	0.75	0.7
Senti4SD_based	Positive	0.83	**0.92**	0.87	**0.91**	0.91	**0.91**	0.3	0.81	0.43
	Negative	0.57	**0.8**	0.67	**1**	0.78	**0.87**	0.42	0.8	0.55
	Neutral	0.89	0.7	0.78	0.79	**0.96**	**0.87**	**0.93**	0.51	0.66
	Micro-avg.	0.78	0.78	0.78	**0.88**	**0.88**	**0.88**	0.59	0.59	0.59
	Macro-avg.	0.76	0.81	0.77	**0.9**	**0.88**	**0.88**	0.55	0.71	0.55
SentiCR_based	Positive	0.6	**0.82**	0.7	**0.74**	0.72	**0.73**	0.73	0.7	0.72
	Negative	0.52	0.4	0.45	0.67	**0.46**	**0.55**	**0.91**	0.25	0.4
	Neutral	**0.82**	0.76	**0.79**	0.76	0.83	**0.79**	0.53	**0.93**	0.67
	Micro-avg.	0.73	0.73	0.73	**0.75**	**0.75**	**0.75**	0.63	0.63	0.63
	Macro-avg.	0.65	0.66	0.64	**0.72**	**0.67**	**0.69**	**0.72**	0.63	0.6
Random	Positive	0.27	**0.95**	0.42	**0.83**	0.89	**0.86**	0.05	0.44	0.09
	Negative	0.11	0.15	0.13	**0.58**	**0.76**	**0.66**	0.05	0.14	0.08
	Neutral	0.95	0.75	0.84	**0.97**	**0.93**	**0.95**	0.96	0.71	0.81
	Micro-avg.	0.74	0.74	0.74	**0.92**	**0.92**	**0.92**	0.69	0.69	0.69
	Macro-avg.	0.44	0.62	0.46	**0.8**	**0.86**	**0.82**	0.35	0.43	0.33
	Overall Micro-avg.	0.76	0.76	0.76	**0.83**	**0.83**	**0.83**	0.65	0.65	0.65
	Overall Macro-avg.	0.66	0.72	0.66	**0.79**	**0.79**	**0.79**	0.57	0.63	0.54

in emails have been both unprofessional and uncalled for,"]. Torvalds stepped down because people where complaining about his lack of care sentiments in his communications which has hurt some contributors and may have driven some away from working in kernel development altogether *[" I'm going to take time off and get some assistance on how to understand people's emotions and respond appropriately".].* Empirical evidence of the effect of expressed sentiments contained into comments on code reviews could help developers pay more attention to the way they comment on other's work, especially in a context of virtual communities such as Github characterized by multicultural contributors.

We are also interested in understanding how the expressed sentiments of contributors evolve over time as they gain in seniority within a project. There has been research examining OSS contributors' involvement over time [26], in particular, researchers pointed out that empirical analyses that mix the two groups will likely yield invalid results. Surprisingly little research has examined the evolution of text-based sentiments when contributors gain reputation (*i.e.,* belong to the core team leading the project). Reviewer's sentiment may wax and wane as project progresses. We thus derive the following research question.

- **RQ2.1: How are positive and negative sentiments expressed in code reviews?**

Previous research [2, 33, 36] that observed significant presence of sentiments and emotions in code reviews and issue comments . Therefore, before analyzing the relationship between sentiments expressed in code reviews and code review outcomes, it is important to learn whether sentiments are also prevalent in our dataset of code review comments. Given that developers can express as well as seek opinions in diverse development scenarios [47, 48], their expression of opinions in code review comments may be influenced by diverse development needs and situations. Therefore, it is necessary to learn how developers expressed those sentiments and what could have triggered the developers to express those opinions.

- **RQ2.2: How do the prevalence of Expressed Sentiments of Reviewers Evolve Over Time?**

We are interested in analyzing potential differences in expressed sentiments between core and peripheral contributors. Core members are those developers that contribute intensively and sustainably to the OSS project, and thus, lead the community, while peripheral ones are occasional contributors with less frequent commits. Our main purpose is to study the correlation between a gain of contributors reputation and the nature of sentiments they express within reviews comments. We hypothesize that newcomers try to imitate contributors with a certain reputation, which might affect the culture of commenting. Hence, we formulate the following research questions:

- **RQ2.3: Do Core and Peripheral Contributors Express Different types of Sentiment According to their Position in a Collaborative Social Network Graph?**

RQ2.1: How are positive and negative sentiments expressed in code reviews?
• **Approach:** Our dataset contains more than 4.4 million comments on code reviews regarding four long-lived and well known OSS projects: *Openstack,*

Eclipse, Android, and *LibreOffice*. The distribution of comments is shown in Table 5. Next, we conducted a sentiment analysis on comments using natural language processing techniques. We used *Senti4SD* tool, a fully-automated algorithm, to compute the sentiment score for each comment on each review. To determine whether sentiment scores are consistent in the projects, we calculate the skewness and kurtosis of the sentiment scores. The skewness of a distribution captures the level of symmetry in terms of mean and median. For instance, a negative skew means that the overall reviews are towards negativity, while a positive skew means that the reviewers overall express more positivity. Kurtosis explains the shape of the distribution (univariate normal distribution is 3). A kurtosis lower than 3 means that the reviewers have a strong consensus, while a kurtosis greater than 3 means a divergence. In order to investigate further the type of sentiments expressed within code review comments, we manually tagged positive (666) and negative (443) comments from the dataset used to answer *RQ1*. To do so, we leveraged on categorization provided by Tourani et al. [49], which categorized positive sentiments into six categories and negative ones into four categories as described in Table 6. Each comment was manually categorized by two raters, the agreement between the two coders, measured using Cohen's kappa, was 61% for positive comments and 65% for negative comments.

• **Results: 8.31% of comments were reported as positive** (score = 1) in the Eclipse project. **While 89.92% of comments were neutral** (score = 0) and **1.77% were negative sentiments** (score = -1). Table 5 summarizes the results of sentiment computation for the studied projects and provide some descriptive statistics, *i.e.,* the mean, standard deviation, kurtosis, and skewness. We noticed relatively similar distributions concerning the proportion of expressed sentiment within comments across the four projects. Neutral comments are the most present (**83.81%**) which confirms the results of previous studies [32]. The large amount of neutral sentiments can be mainly explained by the presence of technical vocabulary within comments. For instance, in Eclipse project, over 153 thousand comments, 89.92% was reported as neutral.

We found that **13.94%** of comments related to all projects were positive (*e.g., "Thanks for the most excellent review. :)"*), while around **2.24%** of comments were identified as negative (*e.g.,*"Horrible :(").

Eclipse is the only project with a Kurtosis value greater than 3 which suggests that sentiment are diverse among the contributors while Openstack, Android, and LibreOffice have a Kurtosis slightly less than 3, meaning that

Table 5: Distributions of Sentiments in Reviews

Project	Positive	Neutral	Negative	Mean	SD	Kurtosis	Skewness
Openstack	16.27 %	81.04 %	2.68 %	0.13	0.41	1.63	0.89
Eclipse	8.31 %	89.92 %	1.77 %	0.06	0.31	6.25	1.53
Android	14.06 %	84.09 %	1.86 %	0.12	0.37	2.43	1.22
LibreOffice	17.14 %	80.21 %	2.65 %	0.14	0.42	1.39	0.87

reviewers have a strong consensus on sentiment expression in source code reviews.

Overall, results reveal that the distribution is highly positively skewed for Eclipse and Android while moderately skewed for Openstack and LibreOffice.

Manual annotation revealed that 'Friendly Interaction' is the most prevalent category of positive sentiments with a proportion of 37.1% as reported in Table 6. This means that, to a large extent, 37.1% of positive interactions between the community' members are guided with respect and positive attitudes. 'Satisfactory Opinion' and 'Announcement' count respectively for 19.6% and 15.5%. While the most common category of negative sentiment is 'Uncomfortable Situation' with a percentage of 62.30%. This means that 62.30% of negative comments express strong pressures such as time constraints that could overwhelm them, confusion about inexplicable behavior of the software system, or concerns about risks and fears. 'Unsatisfied Opinion', 'Aggression', and 'Sadness' count respectively for 16.03%, 15.35% and 6.32%.

As stated by [49], well-mannered interactions with a positive undertone might lead to a higher productivity. Our RQ3 will investigate the impact of expressed sentiments into comments on the duration and the outcome of a source code reviews.

Table 6: Categorization of sentiments within code review comments.

Sentiment	Category	Example	Total	ratio
Positive Sentiment	Satisfactory Opinion	Thank you Andrey!I really appreciate that you took the time to check that, and I'm glad to hear that performance is now okay :)	130	19.6%
	Friendly Interaction	Works fine no issues	247	37.1%
	Explicit Signals	Restored I'll revive this, it makes the debug info analysis much more pleasant.	82	12.3%
	Announcement	I'm sure you see how having all your patches in one chain helps for sanity	103	15.5%
	Socializing	My pleasure :).	59	8.9%
	Curiosity	Before the change, the tests were working fine on the command line.	44	6;6%
Negative Sentiment	Unsatisfied Opinion	Forgot to publish these. Sorry!	71	16.03%
	Aggression	PS. I hate this change and this API.	68	15.35%
	Uncomfortable Situation	I'm sorry but your approach looks like overkill to me.	276	62.30%
	Sadness	I really dislike this patch and "I would prefer that you didn't submit this" but I don't know if it's a valid reason to -1 it.	28	6.32%
Neutral Sentiment	-	-	1111	

RQ2.1 How are positive and negative sentiments expressed in code reviews?

Open source software developers do express sentiments when they are reviewing each other source code. A percentage of 13.94% of comments related to all projects were positive, while around 2.24% of comments were identified as negative. Also, 'Friendly Interaction' is the most prevalent category of positive sentiments with a proportion of 37.1%, 'Uncomfortable Situation' is the most common category of negative sentiment with a percentage of 62.30%.

RQ2.2: How do the prevalence of Expressed Sentiments of Reviewers Evolve Over Time?

• **Approach:** To answer this research question, we proceeded as follows. First, we examined the sentiment evolution of top 5% contributors for each project during the complete time period under study. We pick the top 5% to ensure that we have the most active contributors without any discontinuity in the review activity. In total, over the four studied projects, we analyzed the evolution of expressed sentiments of the top 5% (484 out of 9680 contributors) who have created 1,493,224 comments (33.73% of total comments). After zooming on this group of contributors, we explored manually in details the time series of the top five contributors for one project, which produced 7,184 comments (0.16% of the total comments) to ground sentiment evolution patterns. We focused only on 5 members because of the high cost of the analysis.

• **Results:** We observed a trend toward neutral sentiments correlated with the progression of contributors toward the core team. **The more a contributor gains reputation, the more he is likely to express neutral sentiments**. Figure 4 shows the average of sentiment evolution per month of top 5% contributors (*i.e.,* reviewers). However, we cannot conjecture that this trend towards neutral sentiments is due to a gain of reputation by contributors. It could also be simply due to cultural changes in the studied projects. Further analysis are necessary to better understand the evolution of developers' sentiments in OSS projects.

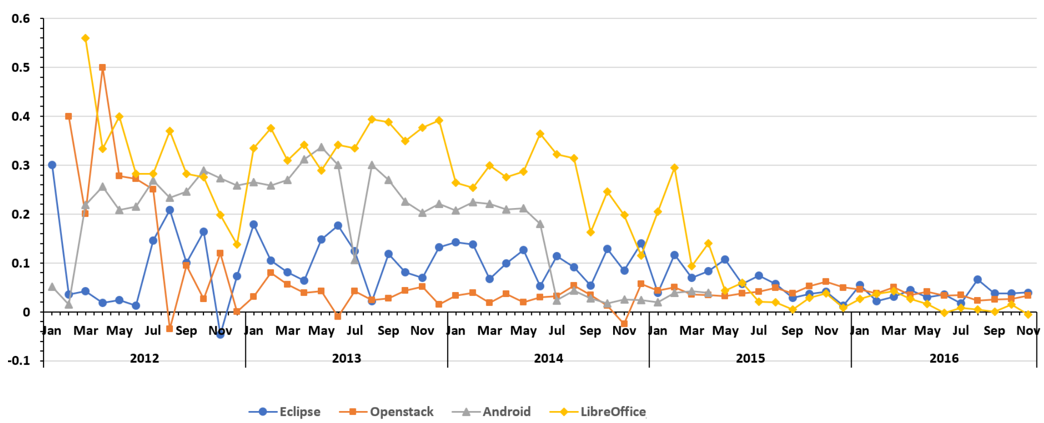

Figure 4: Average Sentiment Evolution per Month of the Top 5% Contributors.

In order to get more insights on sentiment average evolution, we monitor the top 5 core contributors for the Eclipse project. We choose the Eclipse project because it is intensively studied in the literature. As one can see in Figure 5, the sentiment averages vary significantly over years, decreasing from positive towards neutral. The sentiments of the top 5 reviewers in Eclipse decreased to neutral over time. As mentioned earlier, an interesting future qualitative research would be surveying the behavior of the most productive contributors.

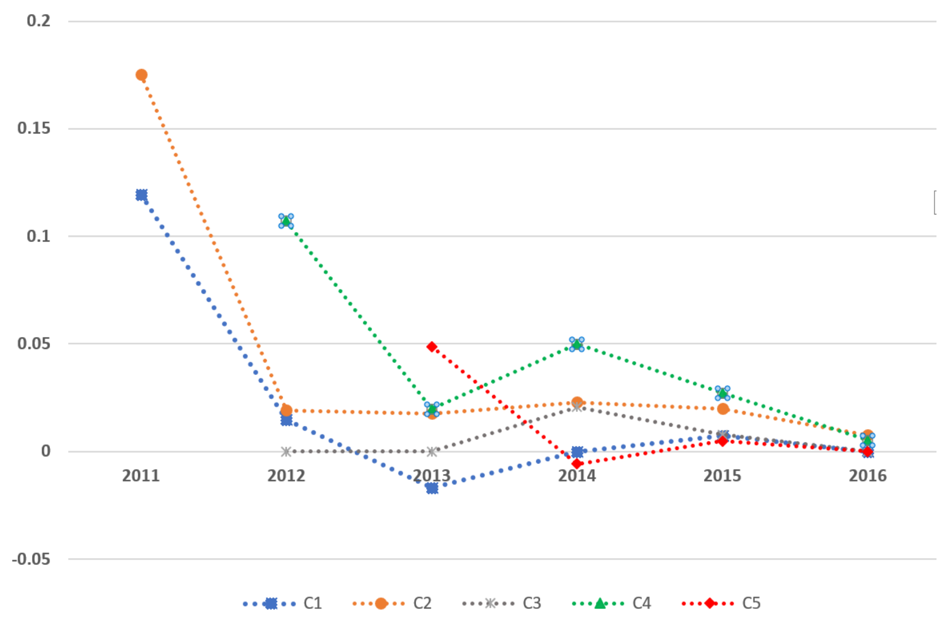

Figure 5: Evolution of the sentiment average (per year) for the top 5 core contributors of the Eclipse project.

RQ2.2 How do the prevalence of Expressed Sentiments of Reviewers Evolve Over Time?

Sentiments expressed by code review contributors tend to be neutral as they progress from the status of newcomer in an OSS project to the status of core team contributor.

RQ2.3: Do Core and Peripheral Contributors Express Different Types of Sentiment According to their Position in a Collaborative Social Network Graph?

• **Approach.** To answer **RQ2** and the sub research questions we need to build Social Networks for each project in order to detect Core and Peripheral contributors. To do so, we calculated the number of interactions between each pair of developers in each project. Then, we generated our social network graphs as undirected, weighted graphs where nodes represent developers and edge weights represent the amount of co-edited files by those contributors. Finally, to locate core and peripheral members we followed the same approach described in [50]. We used the Kmeans clustering method based on SNA centrality measures. Centrality measures used for this approach are : Degree centrality, Betweenness centrality, Closeness centrality, Eigenvector centrality, Eccentricity and PageRank. Each metric calculates centrality in a different way and has a different interpretation of a central node [24].

Concretely, we used the Python package *NetworkX* to calculate the six centrality measures for each node in each graph. Then, we used the R implementation of the Kmeans clustering algorithm to partition the nodes into core and peripheral groups based on the six centrality scores. K-means groups the project contributors into two mutually exclusive clusters in which each contributor belongs to the cluster with the nearest mean (measured using different centrality measures). K-means treats each contributor as an object having a location in space. It finds a partition in which objects within each cluster are as close to each other as possible and as far from objects in other clusters as possible. We used the kmeans()[13] function within R with default configuration options to identify core and peripheral contributors. Table 7 provides a description of the core-periphery partitions obtained for the four projects in this study, alongside with the goodness which is the between_SS / total_SS values provided as a result when using kmeans() of the classification.

Table 7: Core-Periphery distribution in studied Projects

Projects	Size of the Core	Size of the Periphery	Goodness (%)
Openstack	1,081	4,921	82.6
Eclipse	121	628	82.3
Android	189	2,295	84.2
LibreOffice	45	437	85.3

Figure 6 shows the social network structure of the Openstack project as generated by Cytoscape[14], a tool for networks' visualization. Core developers (shown in yellow) represents a small set of contributors (between 4.55% and 12.14%, for the studied projects) who have generally been involved with the OSS project for a relatively long time and are making significant contributions to guide the development and evolution of the project. Peripheral developers (shown in red) are a larger set of contributors whom occasionally contribute to the project, mostly interacting with core developers, and rarely interacting with each other. To enhance readability of OpenStack graph, we

[13] https://stat.ethz.ch/R-manual/R-devel/library/stats/html/kmeans.html

[14] `http://www.cytoscape.org/`

removed the low-weight degrees (weight $\prec$ 5) and isolated nodes.

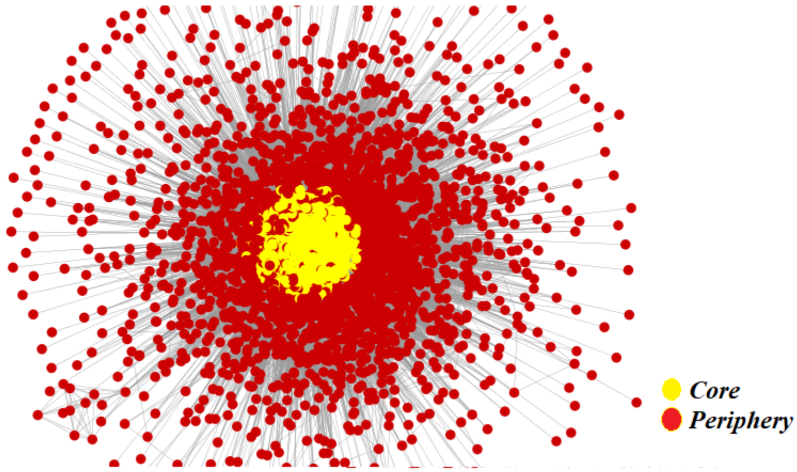

Figure 6: Code Review Social Network Diagram of Eclipse

After segregating core and peripheral contributors, a sentiment score average for each contributor has been calculated based on the sentiment score of all the comments he made. Next, using Mann-Whitney U test [41], we compared the distributions of sentiment averages between groups of core and peripheral contributors. The test is applied following the commonly used confidence level of 95% (*i.e.,* $\alpha \prec$0.05). Since we performed more than one comparison on the same dataset, to mitigate the risks of obtaining false positive results, we use Bonferroni correction [41] to control the familywise error rate. Concretely, we calculated the adjusted p-value, which is multiplied by the number of comparisons. Whenever we obtained statistically significant differences between groups, we computed the Cliff's Delta effect size [41] to measure the magnitude of this difference.

• **Result.** Figure 7 shows the comparison of averages of sentiments between core and peripheral contributors. For the four studied projects, the distribution of sentiment averages ranges between [-1, 1]. The Mann-Whitney test revealed a significant difference in the distribution of sentiment average of core and peripheral contributors. However, the effect size is small, except for the LibreOffice project where it is medium, as reported in Table 8.

Surprisingly, we observed that the peripheral contributors in all four projects have clearly more outliers - *i.e.,* both positive and negative- com-

Table 8: Mann-Whitney U test results

Project	U	p-value	Effect Size
Openstack	4,115,100	2.2e-16	small (0.26)
Eclipse	47,683	1.0e-06	small (0.26)
Android	536,920	2.2e-16	small (0.27)
LibreOffice	22,700	8.48e-12	medium (0.47)

pared to core ones whom sentiments remain concentrated around Neutral emotions (*i.e.,* value equal to zero). We hypothesize that the outliers segment are people participating by a single or a small amount of comments, which impacts the values of averages, whereas Core developers remain neutral while they comment on the source code revisions.

> RQ2.3 Do Core and Peripheral Contributors Express Different Types of Sentiments According to their position in the review network?
>
> Open source contributors do express different sentiments depending on the position within the peer review collaborative social network. Peripheral contributors in the four projects clearly have more outliers in expressing positive and negative sentiments, while Core developers remain neutral when commenting on source code revisions.

RQ3. How do the presence of sentiments in code reviews correlate with the outcome of the reviews?

Motivation. Code review is an essential practice to ensure the long-term quality of the code base. This modern practice could be influenced by expressed sentiment within contributors' comments. Intuitively, positive sentiments may improve the contributors mood, while negative ones may prove detrimental to their morale. Such change in morale can then impact both the time taken and the outcome of the review process. In particular, it is important to know how expressed sentiments can impact code review practices along the following two dimensions: *(1)* Code Review Time, and *(2)* Code Review Outcome. The duration of a source code review is an important factor for a software organization productivity [51]. We pose the following question:

- **RQ3.1 How do the sentiments expressed in the reviews correlate with the duration and the outcome of a review compared**

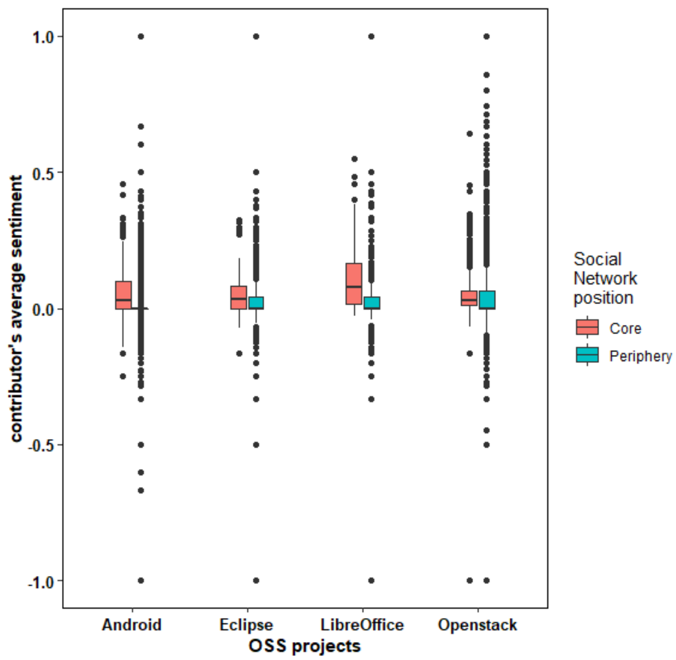

Figure 7: Comparing Sentiments Between Core and Peripheral Contributors.

to the reviews with no sentiments?

When a code review takes much longer than expected, the release of the software and the team productivity can suffer. A number of factors can contribute to such longer time, such as the absence or leave of the core developer responsible for the particular module related to the review or the change in priority. Another mitigating factor could be the presence of *controversy* in the review comments. Intuitively, a feature may be controversial if its code review attracts positive and negative comments almost equally. An empirical understanding of the extent to which such controversies can impact the code review outcome can offer a gain of awareness regarding positive/negative impact of code review practices. As a practical implication, we can motivate a new feature within Gerrit to proactively warn contributors involved in a review team about a risk of delaying the review due to controversies. We thus derive the following research question.

- **RQ3.2. Does the presence of controversies in the code reviews**

offers valuable insights into the outcome of those code reviews compared to the reviews with non-controversial comments?

We are investigating outlier reviews (that took a very long time) in order to determine whether the presence of controversial sentiments in their comments could be the root cause of increased review time.

- **RQ3.3. Do sentiments expressed by core contributors impact the review outcome differently than those expressed by peripheral contributors?**

In RQ2, we observed that core and peripheral contributors express different types of sentiments. Since the contribution of core and peripheral contributors in reviews activities are likely different, *i.e.,* core contributors are expected to be involved more closely than peripheral contributors in code reviews. We are interested in examining whether sentiments expressed by these two groups of contributors also affect the review process differently. In the following we answer three sub questions.

RQ3.1 How do the sentiments expressed in the reviews correlate with the duration and the outcome of a review compared to the reviews with no sentiments?

• **Approach.** An overview of review time distribution in studied projects, pointed out that the slowest review took hundreds of days whereas the median review was less than one day. To avoid bias due to skewed distributions, we used Tukey's outliers detection methods [52]. A review time is considered as outlier if it is above an *Upper limit*. Tukey's define this limit based on the Lower and Upper quartiles [Q1, Q3] (*i.e.,* respectively the 25th and the 75th percentiles of data distribution) such as:

$$Upper\ limit = Q3 + 1.5 * IQR \tag{1}$$

Where inter-quartile range (IQR) is the interval between Q1 and Q3. Tukey's method applied on review time distribution detected a distinct *Upper limit* days for each project (13.86 for Eclipse, 10.02 for Android, 11.04 for Openstack and 6.52 for LibreOffice). The new dataset contains a total of 114,546 reviews.

To assess the influence of positive or negative sentiments on the duration of a code review, we used the **Propensity Score Matching** (PSM)

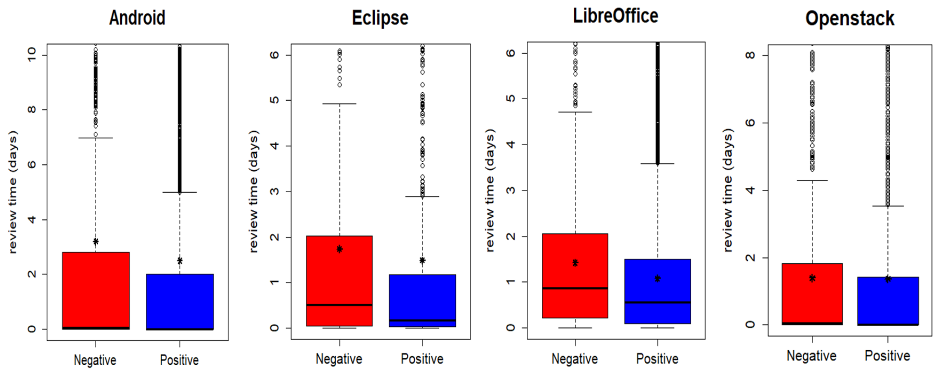

Figure 8: Box-plot of the Duration of Reviews (* : mean).

method [22], as described in Section 4.3. For practical applications, we compare only the reviews that are logically comparable in terms of technical characteristics: *(1) amount of comments for the review; (2) count of patch-Sets, (3) number of edited files,(4) Distinct involved Contributors, (5) and churn* (*i.e.,* sum of inserted and deleted lines of code to measure how large the change is). Also, to assess the influence of sentiments on the reviews' outcome, we mapped the sentiment summary of each review (Positive or Negative) with its final status (Merged or Abandoned).

• **Results.** Comparing a homogeneous group of reviews (obtained through PSM) reveals that positive reviews **took less time** to be closed than negative ones, as depicted in Figure 8. **Negative reviews required a supplementary time of 1.32 day on average to be closed than positive ones.** In other words, the average of durations for positive reviews is less than the average for negative reviews.

Also, as shown in Figure 8, positive reviews not only have the minimum median review time, but also, they have the lowest maximum number of days needed to be closed, compared to negative reviews. For instance, in the Eclipse project, positive reviews last a maximum of 2.89 days, while reviews containing negative comments took approximately 5 days of review. Also, the Mann-Whitney test revealed a significant difference in the distribution of reviews fixing times between positives and negatives reviews with a small

Table 9: The p-value and the effect size of review times in positive vs negative reviews

Project	p-value	Effect Size
Openstack	1.6e-4	small (0.02)
Eclipse	0.2e-4	small (0.09)
LibreOffice	2.8e-11	small (0.17)
Android	1.6e-4	small (0.08)

effect size[15]) for all studied projects (seeTable 9).

Figure 9 shows mapping results of reviews types (Positive or Negative) with the final status of the review (Merged or Abandoned). For each project, the ratio values presents the distribution percentage of positive and negative reviews within merged review (first bar) and abandoned ones (second bar). Results show that, not only does the sentiment expressed by developers affect the duration of code review, but it also affects the outcome. For instance, in Eclipse project, over 93% of successfully merged reviews were tagged as positive, while 55% out of all abandoned reviews have negative sentiments into their comments.

RQ3.1 How do the sentiments expressed in the reviews correlate with the duration and the outcome of a review compared to the reviews with no sentiments?

The presence of positive sentiments in comments related to source code reviews seems to contribute to reducing the review time by an average of 0.4 day. It also seems to affect code reviews outcomes.

RQ3.2. Does the presence of controversies in the code reviews offers valuable insights into the outcome of those code reviews compared to the reviews with non-controversial comments?

• **Approach.** In the previews questions, we analyzed only reviews that took less than the Upper limit before being accepted or abandoned. In this research question, we examine reviews that took a very long time; *i.e.,* more than identified threshold. Our goal is to investigate whether the presence of

[15] *p*-value and effect size are measured using Mann-Whitney U test and the Cliff's Delta effect size as explained in RQ2.2

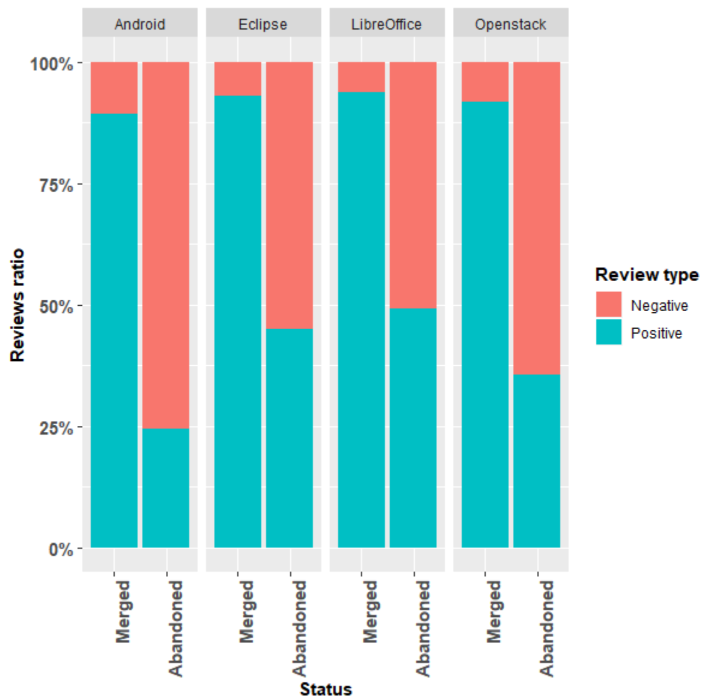

Figure 9: Ratio of Positive vs. Negative Reviews Regarding Reviews' Outcome.

controversy in reviews discussion is the root cause of the long delays. The Merriam-Webster Dictionary defines controversy as a "strong disagreement about something among a large group of people". In our context, we classify a review as *controversial* if the discussions about the submitted source code contain controversial comments. We compute the degree of controversy using controversialMix [53], which is a score that estimates how many mixed positive and negative comments are in a review discussion.

$$\text{ControversialMix} = \frac{(Min((|Pos|, |Neg|)))}{(Max((|Pos|, |Neg|)))} \frac{(|Pos| + |Neg|)}{(|Neu| + |Pos|, +|Neg|)} \quad (2)$$

Where Pos, Neg and Neu are the sets of comments with positive, negative and neutral polarity.

ControversialMix takes in consideration the amount of positive, negative and neutral comments in order to capture the diversity on expressed sentiments within the same review. Before computing controversialMix we did

Table 10: Distribution of Controversial and Non Controversial comments within outliers reviews (Yes =controversial, No= Not controversial)

Project	Controversial	#Reviews	Avg_review _time(days)	P_value
Android	No	780	108.48	0.37
	Yes	31	120.38	
Eclipse	No	185	126.95	0.7
	Yes	4	132.46	
LibreOffice	No	442	44.18	0.01
	Yes	34	50.51	
Openstack	No	10,201	61.82	0.17
	Yes	95	67.30	

some data prepossessing by discarding : reviews with only one comment; reviews where all comments have the same tag (negative, positive, neutral) and reviews threads that have only positive or negative comments. Finally, a review is tagged as controversial if controversialMix ≥ 0.5.

• **Results.** Table 10 shows the distribution of controversial reviews and the average review time needed to fix controversial and non controversial reviews.

Wilcoxon test applied on controversial and non-controversial reviews reveals that results about average review time (days) were significant for only one project: LibreOffice with a p-value $= 0.01$. For this particular project, one can see that controversial reviews required in average more days to be closed (+6.33 days). However, the limited amount of controversy identified for Eclipse, Android, Openstack respectively (4, 31, 95) compared to the amount of controversial reviews found in LibreOffice as shown in Table 10, could explain the non significant result obtained from the wilcoxon test on these projects. Consequently, we were not able to confirm this finding due to the lack of data.

RQ3.2. Does the presence of controversies in the code reviews offers valuable insights into the outcome of those code reviews compared to the reviews with non-controversial comments?

Controversy significantly increased the time taken to review code in the LibreOffice project (44.18 to 50.51). Unfortunately, we did not have enough controversial reviews in the other projects to confirm our finding.

RQ3.3 How do the sentiments expressed by the core vs the peripheral contributors correlate with the outcomes of the code reviews?

• **Approach.** We create two buckets for each project, one for each type of contributors (*i.e.,* core and peripheral). For each class of contributors, we create three polarity buckets, labeled as positive, negative, and neutral. For instance, the positive bucket contains all the review times (in days) of positive reviews. The negative bucket contains all the review times (in days) of negative reviews. The neutral bucket contains all the review times (in days) of neutral reviews. In each of these buckets we excluded: *(1)* reviews that took less than one day, *(2)* reviews that took more than the thresholds for each project that we determined using Tukey's outliers detection algorithm (see RQ3.1). Intuitively, from a productivity perspective, it is useless to analyze the impact of sentiments for a review that took less than a day (because it is already an impressive performance).

We then divide each bucket into two further buckets: *(1)* **Mixed.** We put a review in this bucket if it has sentiments expressed by both core and peripheral contributors. *(2)* **Exclusive.** We put a review in this bucket, if it has sentiments expressed by either the core or the peripheral contributors, but not by both in the same review. We compare the opinion impact of core versus peripheral contributors for the reviews in the 'Exclusive' bucket for each project.

• **Results.** In Table 11, we show the summary statistics of the review time (in days) taken when the core or peripheral contributors offered positive or negative reviews. The 'Neutral' column under each contributor type shows the time taken when the contributor offered neutral comments. For all projects, the average review time increased when the peripheral contributors provided negative comments. For all project, the average review time is larger when the contributors provided negative comments than when the core

Table 11: The impact of sentiments expressed by the core vs peripheral contributors on the code review elapsed time (in days)

		Review Time for Overall Sentiment Type			
Project	**Reviewer Type**	**Time Metric**	**Positive**	**Negative**	**Neutral**
Eclipse	**Core**	Average	**4.8**	**5.1**	**4.8**
		Std	3.3	3.2	3.3
		Median	4.0	4.9	3.9
	Peripheral	Average	**4.6**	**4.8**	**4.8**
		Std	3.2	3.4	3.3
		Median	3.7	3.9	3.7
Android	**Core**	Average	**4.1**	**4.3**	**4.0**
		Std	2.4	2.4	2.4
		Median	3.8	4.1	3.3
	Peripheral	Average	**4.1**	**4.9**	**4.0**
		Std	2.5	2.3	2.4
		Median	3.2	3.5	3.5
Libreoffice	**Core**	Average	**3.1**	**3.3**	**3.1**
		Std	1.6	1.7	1.6
		Median	2.9	3.0	2.8
	Peripheral	Average	**3.1**	**4.1**	**3.2**
		Std	1.6	1.6	1.6
		Median	2.9	3.2	2.8
Openstack	**Core**	Average	**4.2**	**4.7**	**4.2**
		Std	2.6	2.8	2.6
		Median	3.9	4.5	3.5
	Peripheral	Average	**4.3**	**4.9**	**4.2**
		Std	2.7	2.3	2.6
		Median	3.5	3.5	3.7

Table 12: The p-value and effect size of review times in the neutral vs non-neutral comments by the core and peripheral contributors

Project	Review Time	Core		Peripheral	
		p-value	δ	**p-value**	δ
Eclipse	Positive vs Neutral	0.40	0.009	0.17	N/A
	Negative vs Neutral	**0.0002**	0.27	0.48	0.151
Android	Positive vs Neutral	**2.18E-07**	0.18	**1.61E-07**	0.17
	Negative vs Neutral	**2.28E-04**	0.31	**0.002**	0.27
Libreoffice	Positive vs Neutral	**9.67E-05**	0.23	**6.79E-05**	0.24
	Negative vs Neutral	**3.11E-01**	0.38	**8.52E-08**	0.46
Openstack	Positive vs Neutral	**7.82E-09**	023	**3.57E-03**	0.27
	Negative vs Neutral	**9.20E-04**	0.42	**6.39E-07**	0.31

contributors provided neutral comments in the reviews. This trend is similar between the core and peripheral contributors, *i.e.,* negative comments from any contributors tend to increase the review time. Except for Eclipse, the increase in average time taken due to the negative comments is *statistically significant* (see Table 12[16]). However, we do not see such impact for positive comments. For peripheral contributors, the impact is more prominent. For only one projects (Eclipse), the review time is less when the peripheral contributors provided positive comments.

Both the core and peripheral contributors seem to equally impact the review time when they provide positive comments in two projects (Android and Libreoffice).

For one project (Eclipse), the positive comments from core contributors seem to impact the review time more than the negative comments from peripheral contributors. For Openstack, the situation is reversed, *i.e.,* the negative comments from the peripheral contributors seem to impact the average review time more than the core contributors.

On average, the review time is much less in the reviews where the peripheral contributors provided positive comments. This finding corroborates

[16] p-value and effect size are measured using Mann-Whitney U test and the Cliff's Delta effect size as explained in RQ3.1

our previous finding that peripheral contributors offer more sentiments in the code reviews, because the core contributors tend to become more neutral over time. Therefore, the happiness of the peripheral contributors seem to be important to reduce the code review time.

RQ3.3 How do the sentiments expressed by the core vs the peripheral contributors correlate with the outcomes of the code reviews?

In all projects except Eclipse, the review times are impacted more by the negative comments from peripheral contributors than the negative comments from the core contributors. In all projects, the review times are longer when the peripheral contributors provided negative comments.

6. Future Possibilities

In all of our studied projects, the reviews with negative sentiments took more time to complete. This observation leads to the question of how we can leverage sentiment analysis to improve *productivity* in a code review process, if the contributors participating in the code reviews can be both the provider and receiver of such negative sentiments. One potential solution would be to design automated sentiment-based monitors that can offer guidance to the contributors. Although such solutions lack authoritativeness, they may nevertheless prove useful to guide the contributors through the different phases of a code review process by mitigating negativity in the review comments. With a view to improve code review outcome and time based on sentiment analysis, we offer the following recommendations by taking cues from our three research questions:

1. Sentiment analysis can be applied to find communities or sub-communities within a project that may be affected by negative comments.

2. Harmful contributors, such as *bullies* can be detected to ensure that they do not impact the review process negatively.

3. Controversial reviews can be identified to warn the project leaders about potential controversial features or communities in a project.

4. Software Bots can be designed to warn the contributors participating in a review when negativity in a review increases.

We now discuss the recommendations below.

6.1. Community-Based Analysis

In RQ3, we built social networks of contributors and observed two major streams of contributors, *core* and *peripheral*. Compared to the peripheral contributors, core members tend to remain with a project for longer time. A deeper understanding of the interactions between the contributors based on social network analysis can offer insights into whether *intrinsic* or *dynamic* sub-communities do exist in modern Gerrit-based code review systems. The identification of such communities can offer several benefits, such as promoting a high-performance community to others, offering guidance to a community that is exchanging negative sentiments but is not productive enough, for examples.

6.2. Bullies Among Contributors

In all the four studied projects, the reviews with negative sentiments took longer time to get accepted. One possible explanation is that a patch with bugs is likely to be viewed negatively and thus will not be accepted or will be iterated until fixed. However, it is not easily explainable why the negative comments from peripheral contributors impacted the review time more than the core contributors. One potential reason could be that the peripheral contributors are mostly novices to the system compared to the core contributors. Therefore, they would have expressed frustrations due to their lack of understanding of the system.

Another possible reason is that there could be *bullies* among the contributors, who try to influence system design and code review outcome using negative comments. Such negative comments can also impact the contributors. Indeed, Màntylà *et al.* [54] observed that emotions expressed in Jira issues can be correlated to the burnouts of the developers. Ortu *et al.* [36] observed in Jira issues that despite being negative, *bullies* are not necessarily more productive than other developers. An understanding of the role of potential *bullies* in code reviews can offer benefits, such as their impact on the code review outcome and productivity. Measures can be taken to detect the *bullies* among the contributors and to remove them from the review process.

6.3. Impact of Controversies

As we observed in *RQ4*, regarding reviews taking a long time, the presence of controversy can increase the review time even more. The analysis of controversy has proved useful in social media, such as to detect fake news [55]. A deeper analysis of the controversial code review comments can offer insights into the specific reasons behind the comments. For example, it may happen that the product feature (for which the patch is provided) may not be well designed, such that the contributors debate during the review process. It may also happen that the feature is not well-received, such that the contributors have different viewpoints on how to improve it. Therefore, measures can be taken to mitigate the controversies and thus improve the code review outcomes.

6.4. Review Sentiment Bot

Bots have been developed to assist in numerous software development activities, such as automatically suggesting an answer from Stack Overflow given a query [56], answering questions about an API from documentation [57], or warning developers in a GitHub project if they post negative comments [58]. We can develop similar bots to automatically warn the contributors in a code review system with the automated detection of negative comments, their prevalence in the controversies and their proliferation by the bullies. As a first step, we can start with the adaptation of Github sentiment bot [58] for code reviews.

6.5. Gender and cultural aspects

We have investigated gender and cultural aspects bias concerns by defining the following null hypothesis:
H0: There is no significant difference in text-based sentiment between male and female contributors.
H1: There is no significant difference in text-based sentiment between contributors from different countries, which have different language and cultures.

Difference between genders - Female and Male - may reveal interesting facts under appropriate analysis. Indeed, recent studies discussed gender bias regarding productivity, in terms of commits, in OSS projects [59, 60]. Moreover, Terrell et al. [61] reported that when new female contributors are identifiable, they have 12% lower chance of getting their pull request accepted than other females whose gender was not identifiable from their profiles. Hence, we are interested in this work to know if there is an association between developers' genders and their expressed sentiments. More specifically, we formulate the following research questions: **Are females' contributors more likely to be positive/neutral/negative**

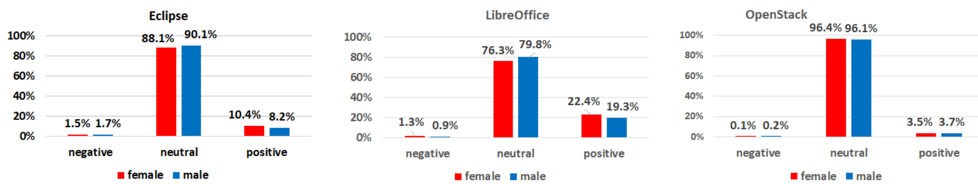

Figure 10: Distribution of Sentiments According to Gender.

than males? Is the proportion of females that express negative sentiments the same as the proportion of males? To answer these questions, we segregated contributors according to their gender. We used the NamSor[17] API to classify contributors into binary gender given personal names, country of origin, and ethnicity. This API infers gender from the combination of first name, surname, and information of the country. We found that 6.8% of Eclipse contributors were females and 88.9% were male, and 4.4% unknown. LibreOffice respectively (9.4% ; 86.5% ; 4.1%) ; and OpenStack(10.9% ; 79.6% ; 9.5%). Unfortunately, we were not able to resolve genders for the Android project because of (encrypted name and email). Figure 10 shows the distribution of sentiments across gender for three projects.

One first observation is that women and men seem to exhibit the same distribution of sentiments. We performed further statistical analysis to verify how genders differed in their expressed sentiments. Given that the variables do not exhibit a normal distribution, we performed a (non-parametric) Mann-Whitney-Wilcoxon test, with a confidence level of α =0.05. We found that, overall for the three projects, the tests are statistically significant ($p \prec 0.05$, Z statistic of -1.018) and thus we reject our null hypothesis H0. We claim that there is a significant difference in the distribution of text-based sentiment between male and female contributing to OSS projects. This result confirms previous findings by Paul et al. [34].

We also investigated the impact of the country origin for the top 5% core contributors within the three projects aiming to investigate the impact of the first language and cultural aspects of these contributors on code reviews. Figure 11 shows the geographical distribution of the top 5% contributors. We performed a Kruskal-Wallis statistical test to verify whether samples (i.e., different countries) have the same distribution of expressed sentiments.

[17] https://www.namsor.com/

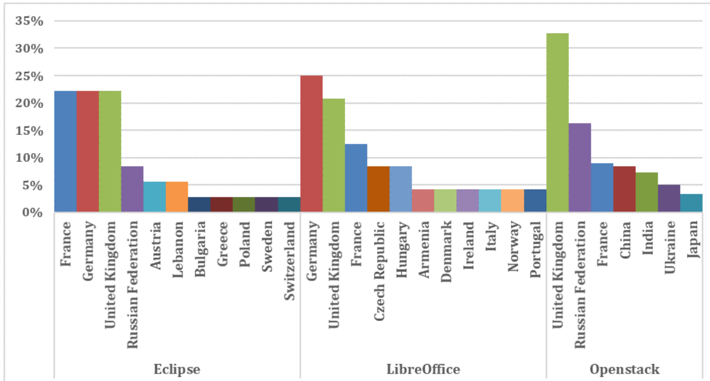

Figure 11: Countries distribution for top 5% contributors.

Kruskal-Wallis test results reveal that distribution differences are statistically significant (p_value $\prec$ 2.2e-16), we reject our null hypothesis H1 and state that there is a statisticaly significant difference in expressed sentiments according to the country of origin.

However, studying the effect of gender cultural aspects on code reviews are beyond the scope of this study. We will address this concern in future work.

7. Threats to Validity

Threats to construct validity are mainly related to the accuracy of the tool used for sentiment analysis. We strengthen our sampling approach for manual annotation by using opportunistic sampling. The authors manually examined 2,220 comments. In general, we observed that the sentiments expressed in code reviews are easy to analyze due to the unambiguous nature of text-based sentiments expressed in the code reviews comments, which were understood by both coders with relative ease.

Threats to internal validity concern factors internal to our study that may affect our findings. The primary threat to internal validity in this study relates to project selection. One possible threat is that the retrieved dataset is too small and somehow is not representative enough. We were cautious to choose OSS projects with the following characteristics: *(1)* long-lived projects with dynamic communities around; *(2)* the community uses the review tools Gerrit to carry out code reviews activities. We also paid attention not to violate assumptions of the statistical tests, for example we applied non-parametric tests that do not require making assumptions on the normality of our data set.

In addition, we used propensity score matching [62] to eliminate the bias that could be introduced by technical characteristics. We compared the distribution of estimated propensity score between Positive and Negative reviews in the matched sample and obtained an average of 96% of overlap, which means that we are dealing with a homogeneous data set of reviews based on the observed covariate values. This provides confidence that the observed results are not due to structural differences in the patches (*i.e.,* we are not comparing large patches with small patches, etc.). However other technical characteristics can be considered such as the number of sentences in the comments and code complexity.

Threats to external validity concern the generalization of our findings. In the context of RQ1 we performed 2,220 manual classifications. We are aware that the quality and size of the annotated set may impact the sentiment classification accuracy. While the human raters are knowledgeable in mining software repositories, sentiment analysis and empirical analysis, their judgement may be impacted by the absence of related in-depth information of the studied systems in the dataset, e.g., whether the reviewers in those studied systems exhibited any latent communities as reported by Bird et al. [63]. Furthermore, our study involves only four projects. Thus, we should recognize that our conclusions may not be generalizable to other systems. We are also aware that the context of each project including the technical complexity and organization are important factors that can limit generalization. However, these projects are among the most studied projects in the literature and the system's data are publicly available. We also have the opportunity to perform a longitudinal study over more than five years, which mitigates the risks related to cultural aspects. Yet, replication of our work on other open and close source systems is desirable in order to generalize our conclusions.

Threats to reliability validity refers to the degree to which the same data would lead to the same results when the study's design is replicated. Our research aims at investigating expressed sentiment by developers on reviews. Our methodology for data analysis and results are well documented in this paper. The tools are available [39] and our datasets are publicly available online [43]. Also, all the participants of the manual tagging have a background in computer science; we are confident that reviews comments have been interpreted according to the perspective of software engineers. We did not involve raters with a different background, because they may overlook or misinterpret the terms used by developers. However, RQ2.1 reveals that most comments in the dataset have neutral sentiments, while only less than 3% of the comments are negative which may have an impact on our analysis and results. In RQ3.2, we assessed whether reviews with sentiments took a shorter/longer time than the reviews with neutral sentiments. We noticed a low number of negative sentiments, similarly observed in previous studies that used datasets from Stack Overflow (e.g., the Stack Overflow dataset by Lin et al. [8] has around 75% neutral comments). Therefore, although the low number of negative comments may introduce a threat to the generalizability of our results across other systems, our analysis remains applicable to other systems. In addition, we assessed the impact of sentiments on code review outcomes in RQ3.3 by comparing the time taken for reviews with positive comments vs the reviews with negative comments. We found on average 13.94% positive and 2.24% negative comments in the four studied systems.

8. Conclusions

We have analyzed developers' comments on reviews using historical data from four open source projects. We aimed at investigating the influence of text-based expressed sentiments on the code review duration and its outcome. Using the best performing sentiment detection tool, we found that contributors do express sentiments when they are reviewing and commenting each other's code. Also, we investigated the influence of expressed sentiments within developers' comments on the time and outcome of the code review process. We found that expressing positive sentiment when reviewing source code have an influence on reviews duration time; in average it could save 1.32 days on the review completion time. Moreover, our findings indicate that negative comments are likely to increase the proportion of unsuccessful reviews.

From a social network perspective, we used a K-means clustering approach based on SNA centrality measures to discern between core and peripheral contributors. We found that different contributors within the peer review collaboration social network express different sentiments, with core contributors expressing mostly neutral sentiments.

Our work contributes theoretically and empirically to the body of OSS research and has practical implications on sentiment awareness within OSS. We hope that our work will inspire more studies on developing efficient tools to help OSS contributors improve their productivity. As future work, we plan to complement this quantitative study with a qualitative exploration aiming at gaining more understanding of the influence of expressed sentiments on code revision workflow. Also, we plan to investigate the effect of developers' expressed sentiment on contributor's engagement and–or turnover.

References

[1] A. Bosu, J. C. Carver, Impact of peer code review on peer impression formation: A survey, in: Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on, IEEE, pp. 133–142.

[2] A. Murgia, P. Tourani, B. Adams, M. Ortu, Do developers feel emotions? an exploratory analysis of emotions in software artifacts, in: Proceedings of the 11th International Conference on Mining Software Repositories, MSR'14, pp. 262–271.

[3] B. Pang, L. Lee, Opinion mining and sentiment analysis, Foundations and Trends in Information Retrieval 2 (2008) 1–135.

[4] O. Kucuktunc, B. B. Cambazoglu, I. Weber, H. Ferhatosmanoglu, A large-scale sentiment analysis for yahoo! answers, in: Proceedings of the Fifth International Conference on Web Search and Data Mining, WSDM '12, pp. 633–642.

[5] D. Garcia, M. S. Zanetti, F. Schweitzer, The role of emotions in contributors activity: A case study on the gentoo community, in: Cloud and green computing (CGC), 2013 third international conference on, IEEE, pp. 410–417.

[6] O. Baysal, O. Kononenko, R. Holmes, M. Godfrey, The influence of non-technical factors on code review, in: proceedings of the 20th Working Conference on Reverse Engineering, pp. 122–131.

[7] V. Efstathiou, D. Spinellis, Code review comments: Language matters, in: Proceedings of the 40th International Conference on Software Engineering (ICSE'18), p. 4.

[8] B. Lin, F. Zampetti, G. Bavota, M. Di Penta, M. Lanza, R. Oliveto, Sentiment analysis for software engineering: How far can we go?, in:

Proceedings of the 40th International Conference on Software Engineering (ICSE'18), p. 11.

[9] N. Imtiaz, J. Middleton, P. Girouard, E. Murphy-Hill, Sentiment and politeness analysis tools on developer discussions are unreliable, but so are people, in: Proceedings of the 3rd International Workshop on Emotion Awareness in Software Engineering SEmotion'18, pp. 55–61.

[10] N. Novielli, D. Girardi, F. Lanubile, A benchmark study on sentiment analysis for software engineering research, in: Proceedings of the 15th International Conference on Mining Software Repositories, p. 12.

[11] F. Calefato, F. Lanubile, F. Maiorano, N. Novielli, Sentiment polarity detection for software development, Empirical Software Engineering (2017) 31.

[12] T. Ahmed, A. Bosu, A. Iqbal, S. Rahimi, Senticr: A customized sentiment analysis tool for code review interactions, in: Proceedings of the 32nd International Conference on Automated Software Engineering, pp. 106–111.

[13] M. R. Islam, M. F. Zibran, Leveraging automated sentiment analysis in software engineering, in: Proceedings of the 14th International Conference on Mining Software Repositories, MSR '17, pp. 203–214.

[14] M. D. Munezero, C. S. Montero, E. Sutinen, J. Pajunen, Are they different? affect, feeling, emotion, sentiment, and opinion detection in text, IEEE transactions on affective computing 5 (2014) 101–111.

[15] W. G. Parrott, Emotions in Social Psychology, Psychology Press, 2001.

[16] B. Pang, L. Lee, S. Vaithyanathan, Thumbs up? sentiment classification using machine learning techniques, in: Conference on Empirical Methods in Natural Language Processing, pp. 79–86.

[17] B. Liu, Sentiment Analysis and Opinion Mining, Morgan & Claypool Publishers, 2012.

[18] P. Weissgerber, D. Neu, S. Diehl, Small patches get in!, in: Proceedings of the 2008 International Working Conference on Mining Software Repositories, MSR '08, pp. 67–76.

[19] O. Kononenko, O. Baysal, L. Guerrouj, Y. Cao, M. Godfrey, Investigating code review quality: Do people and participation matter?, in: Proceedings on the International Conference on Software Maintenance and Evolution (ICSME'15), pp. 111–120.

[20] M. Beller, A. Bacchelli, A. Zaidman, E. Juergens, Modern code reviews in open-source projects: Which problems do they fix?, in: Proceedings of the 11th Working Conference on Mining Software Repositories, MSR 2014, pp. 202–211.

[21] O. Kononenko, O. Baysal, M. W. Godfrey, Code review quality: How developers see it, in: Proceedings of the 38th International Conference on Software Engineering, ICSE '16, pp. 1028–1038.

[22] A. Thavaneswaran, Propensity score matching in observational studies, Manitoba Center for Health Policy. (2008).

[23] M. E. Newman, The structure and function of complex networks, SIAM review 45 (2003) 167–256.

[24] L. C. Freeman, The development of social network analysis–with an emphasis on recent events, The SAGE handbook of social network analysis 21 (2011) 26–39.

[25] X. Yang, Social network analysis in open source software peer review, in: Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE'14, pp. 820–822.

[26] K. Crowston, K. Wei, Q. Li, J. Howison, Core and periphery in free/libre and open source software team communications, in: Proceedings of the 39th International Conference on System Sciences, pp. 118.1–.

[27] I. Robertson, C. Cooper, Well-being: Productivity and happiness at work, Springer, 2011.

[28] M. Thelwall, K. Buckley, G. Paltoglou, Sentiment strength detection for the social web, Journal of the American Society for Information Science and Technology 61 (2012) 2544–2558.

[29] R. Jongeling, S. Datta, A. Serebrenik, Choosing your weapons: On sentiment analysis tools for software engineering research, in: Software

maintenance and evolution (ICSME), 2015 IEEE international conference on, IEEE, pp. 531–535.

[30] J. Guillory, J. Spiegel, M. Drislane, B. Weiss, W. Donner, J. Hancock, Upset now?: Emotion contagion in distributed groups, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '11, pp. 745–748.

[31] E. Guzman, B. Bruegge, Towards emotional awareness in software development teams, in: Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2013, pp. 671–674.

[32] V. Sinha, A. Lazar, B. Sharif, Analyzing developer sentiment in commit logs, in: Proceedings of the 13th International Conference on Mining Software Repositories, MSR '16, pp. 520–523.

[33] E. Guzman, D. Azócar, Y. Li, Sentiment analysis of commit comments in github: An empirical study, in: Proceedings of the 11th Working Conference on Mining Software Repositories, MSR'14, pp. 352–355.

[34] R. Paul, A. Bosu, K. Z. Sultana, Expressions of sentiments during code reviews: Male vs. female, in: Proceedings of the 16th International Conference on Software Analysis, Evolution and Reengineering SANER'19, pp. 15–26.

[35] I. A. Khan, W.-P. Brinkman, R. M. Hierons, Do moods affect programmers' debug performance?, Cognition, Technology & Work 13 (2011) 245–258.

[36] M. Ortu, B. Adams, G. Destefanis, P. Tourani, M. Marchesi, R. Tonelli, Are bullies more productive?: Empirical study of affectiveness vs. issue fixing time, in: Proceedings of the 12th Working Conference on Mining Software Repositories, MSR'15, pp. 303–313.

[37] G. Destefanis, M. Ortu, S. Counsell, S. Swift, M. Marchesi, R. Tonelli, Software development: do good manners matter?, PeerJ Computer Science 2 (2016) e73.

[38] M. M. Rahman, C. K. Roy, R. G. Kula, Predicting usefulness of code review comments using textual features and developer experience, in:

Proceedings of the 14th International Conference on Mining Software Repositories, MSR '17, pp. 215–226.

[39] X. Yang, R. G. Kula, N. Yoshida, H. Iida, Mining the modern code review repositories: A dataset of people, process and product, in: Proceedings of the 13th International Conference on Mining Software Repositories, ACM, pp. 460–463.

[40] L. Guo, P. Qu, R. Zhang, D. Zhao, H. Wang, R. Liu, B. Mi, H. Yan, S. Dang, Propensity score-matched analysis on the association between pregnancy infections and adverse birth outcomes in rural northwestern china, Scientific reports 8 (2018) 5154.

[41] A. Dmitrienko, G. Molenberghs, C. Chuang-Stein, W. W. Offen, Analysis of clinical trials using SAS: A practical guide, SAS Institute, 2005.

[42] N. Novielli, F. Calefato, F. Lanubile, A gold standard for emotion annotation in stack overflow, in: Proceedings of the 15th International Conference on Mining Software Repositories, MSR '18, pp. 14–17.

[43] I. E. Asri, N. Kerzazi, G. Uddin, F. Khomh, An Empirical Study of Sentiments in Code Reviews (online appendix), `https://https://github.com/ikramElasri/SentiAnalysis_CodeReview`, October 2018 (last accessed).

[44] D. M. Christopher, R. Prabhakar, S. Hinrich, Introduction to information retrieval, An Introduction To Information Retrieval 151 (2008) 5.

[45] B. Bostanci, E. Bostanci, An evaluation of classification algorithms using mc nemar's test, in: Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012), Springer, pp. 15–26.

[46] J. M. George, A. P. Brief, Feeling good-doing good: a conceptual analysis of the mood at work-organizational spontaneity relationship., Psychological bulletin 112 (1992) 310.

[47] G. Uddin, F. Khomh, Mining API Aspects in API Reviews, Technical Report, Technical Report. 10 pages. http://swat. polymtl. ca/data/opinionvalue , 2017.

[48] G. Uddin, O. Baysal, L. Guerrouj, F. Khomh, Understanding how and why developers seek and analyze api-related opinions, IEEE Transactions on Software Engineering (2019).

[49] P. Tourani, Y. Jiang, B. Adams, Monitoring sentiment in open source mailing lists: Exploratory study on the apache ecosystem, in: Proceedings of 24th Annual International Conference on Computer Science and Software Engineering, CASCON '14, IBM Corp., Riverton, NJ, USA, 2014, pp. 34–44.

[50] A. Bosu, J. C. Carver, Impact of developer reputation on code review outcomes in oss projects: An empirical investigation, in: Proceedings of the 8th International Symposium on Empirical Software Engineering and Measurement, ESEM '14, pp. 33:1–33:10.

[51] G. P. Sudhakar, A. Farooq, S. Patnaik, Measuring productivity of software development teams, Journal of Management 7 (2012) 65–75.

[52] J. W. Tukey, Exploratory data analysis, volume 2, 1977.

[53] A.-M. Popescu, M. Pennacchiotti, Detecting controversial events from twitter, in: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM '10, pp. 1873–1876.

[54] M. Mäntylä, B. Adams, G. Destefanis, D. Graziotin, M. Ortu, Mining valence, arousal, and dominance – possibilities for detecting burnout and productivity?, in: Proceedings of the 13th Working Conference on Mining Software Repositories, pp. 247–258.

[55] K. Garimella, G. D. F. Morales, A. Gionis, M. Mathioudakis, Quantifying controversy on social media, Transactions on Social Computing 1 (2018) Article no. 3.

[56] S. Zamanirad, B. Benatallah, M. C. Barukh, F. Casati, Programming bots by synthesizing natural language expressions into api invocations, in: Proceedings of the 32nd International Conference on Automated Software Engineering, pp. 832–837.

[57] Y. Tian, F. Thung, A. Sharma, D. Lo, Apibot: question answering bot for api documentation, in: Proceedings of the 32nd International Conference on Automated Software Engineering, pp. 153–158.

[58] GitHub, Sentiment Bot, https://github.com/apps/sentiment-bot, 18 May 2018 (last accessed).

[59] B. Vasilescu, D. Posnett, B. Ray, M. G. van den Brand, A. Serebrenik, P. Devanbu, V. Filkov, Gender and tenure diversity in github teams, in: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, CHI '15, ACM, New York, NY, USA, 2015, pp. 3789–3798.

[60] C. Mendez, H. S. Padala, Z. Steine-Hanson, C. Hilderbrand, A. Horvath, C. Hill, L. Simpson, N. Patil, A. Sarma, M. Burnett, Open source barriers to entry, revisited: A sociotechnical perspective, in: Proceedings of the 40th International Conference on Software Engineering, ICSE '18, ACM, New York, NY, USA, 2018, pp. 1004–1015.

[61] J. Terrell, A. Kofink, J. Middleton, C. Rainear, E. Murphy-Hill, C. Parnin, J. Stallings, Gender differences and bias in open source: pull request acceptance of women versus men, PeerJ Computer Science 3 (2017) e111.

[62] Z. Luo, J. C. Gardiner, C. J. Bradley, Applying propensity score methods in medical research: pitfalls and prospects, Medical Care Research and Review 67 (2010) 528–554.

[63] C. Bird, D. Pattison, R. D'Souza, V. Filkov, P. Devanbu, Latent social structure in open source projects, in: Proceedings of the 26th ACM SIGSOFT International Symposium on Foundations of software engineering, ACM, pp. 24–35.