Analyzing The Impact Of Feedback In GitHub On The Software Developer's Mood

Mateus Freira, Josemar Caetano, Johnatan Oliveira, Humberto Marques-Neto

Department of Computer Science
Pontifical Catholic University of Minas Gerais (PUC Minas)
Belo Horizonte, Brazil
{mateus.freira, josemar.caetano, johnatan.oliveira}@sga.pucminas.br,
humberto@pucminas.br

Abstract

Software development depends on cooperation between people, and the way it works can define the future of the software project. Developers emotions affect their productivity and way they work, yet there is little information about how developers can influence the mood of each other. As a first step toward understanding how feedback may affect the developers' sentiment, this paper analyzes the mood variations on more than 78k pull requests and 268k pull comments on GitHub. We found that in 31.16% of the cases the developers presented a significant mood variation within one hour when receiving feedback on their pull requests. The variation reduces to 18.16% when evaluating one day before and after the commentary. In software projects with less than 34k lines of code, the number of developers that never contribute again after receiving a negative comment on the first pull request is 10.97%; this number more than doubles to 24.02% when evaluating projects with more than 197k lines of code.

Keywords: Developers emotions, Developers mood, GitHub, Open source, Subjective well-being.

1 Introduction

Humans are one of the most valuable resources in software projects. Besides the technical knowledge needed for a successful project, it is essential to have good teamwork[7]. Staats [15] shows that increasing the teams familiarity decreases the number of defects, reduces budget deviation, and yields a 10% in performance improvement from the clients' perspective. Regarding the importance of keeping the team's members, Voices [17] warned that unhappiness could lead the developers to quit their company/project endeavors. Additionally, emotions can affect positively or negatively, the developers' productivity, creativity, and task quality[4, 6]. Understanding the mood variations of the developers on a software project, as well as the effects of feedback on the team members, is relevant to help the projects leaders to take proactive actions in increasing their team engagement and familiarity and, therefore, improving productivity.

GitHub is a social network and code hosting provider that hosts more than 71 million projects[1] at present. The main GitHub feature is not only the code hosting but also project manager features, such as issues and pull request controls. These features promote discussions among the users, which is related to the reported bug and the requested feature or even connected to a code path that a developer wants to merge in a repository. We collected pull requests, their comments, and the profile of the developers, related to any of those interactions, to perform our research.

Recently, researchers have published several papers, regarding developer sentiment analysis, on GitHub interactions[5, 7, 8, 9, 12, 14]. However, what causes positive or negative variances in the developers sentiment, as well as the duration of the variation, is not entirely understood yet; many external influencing factors remain unexplored.

Our study contributes to a better understanding of developers' mood variations by analyzing the developers at the time they are submitting pull requests to a repository, and how other developers' comments may influence their mood. We highlight that we *cannot* establish a strong causal relationship between a comment and a developers' mood variation, as events outside of our data set might have influenced such variation. We empirically analyzed the 100 most popular java projects on GitHub and their more than 226k pull request comments. We applied SentiStrength [16] to calculate the sentiment expressed in each pull request and comment, and we then calculated the subjective well-being (SWB) [3] to obtain the mood variations before and after receiving a feedback from another developer. Finally, we analyzed the SWB to understand how the feedback may affect the developers' mood.

The rest of this paper is outlined as follows. We first present the techniques we used to perform the sentiment analysis and to calculate the sentiment influences in Section 2. We show the experiment design in Section 3. In Section 4 we present the results of our analysis, We present the related work in the Section 5. and finally, we conclude and present the plans for future works in Section 6.

[1] https://github.com/about

DOI reference number: 10.18293/SEKE2018-153

2 Background

Reminding that our present goal is to investigate developers' mood variation when receiving feedback from others, we present an overview of the central concepts we have used to hit our goals. We first introduce the GitHub concepts, then we present the sentiment analysis approach we have used, then we show the changes we have made on the tool's dictionary to better address software engineering texts, and finally, we present the sentiment state and change measurement metrics we have used.

GitHub: Here we present some GitHub concepts that will help gain a better understanding of the paper. Next, we pick the features that relate the most to the present work.

The Repository represents the project itself; it contains the code, documentation, and also aggregates all the project interactions (issues, pull requests and commits). Repositories can be private or public; however, in this paper, all the repositories we have used are public, and therefore, all their information is available on the GitHub API [2].

A pull request is a proposed code change to a repository submitted by a developer. Once a pull request is open the project members can comment and either accept or reject it. If the pull request is accepted, the code integrates them into the repository code; if it is rejected, it is discarded. Pull requests have their discussion forum where the developers can comment on the changes or ask for improvements or changes before merging it. A pull request can contain one or many commits [2].

A commit consists of a change in one or many files, enabling developers to track the changes they have made. Commits usually contain a message with a short explanation about the change that it contains [2].

Author association is the association between the developers who are commenting or opening a pull request, and it represents the role of the developer in the repository. The author association's possible values are: collaborator (has been invited to the repository but has not commited anything yet), contributor(has been invited and has at least one commit), member (is a part of organization that owns the repository on GitHub), none(has no relationship with the repository), and owner (owner of the repository), we limited this list to the values that are present in our data set [1].

GitHub pull request flow consists of some developer (any GitHub user) with or without a relationship with the repository, who wants to integrate a code change to the code base. Let's suppose there is a bug and a developer wants to fix it, he/she needs to: 1) clone the project, 2) create a branch locally, 3) commit the changes to solve the bug, and 4) submit a pull request to the central repository. Figure 1 presents this process. After opening the pull request, other developers can comment on it, asking for changes or endorsing the changes made. That is why these comments often become discussions among developers relating to the code change submitted. For that matter, the most critical interactions for us are after opening the pull request, when the discussions start. At this point, we compare the developer comments, before and after another developer's comment, to understand if the comment caused any mood variation on the developer who wanted to integrate his/hers code to the repository.

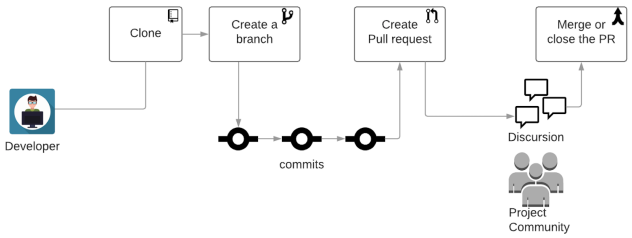

Figure 1: GitHub Pull request flow.

Sentiment analysis: Sentiment analysis is a common task when evaluating social network interactions. Ribeiro et al. [11] presented benchmark testing 24 sentiment analysis tools in 18 labeled tweet data sets. Their results showed that the SentiStrength tool [16] presented the best results in most of the datasets. Additionally, the previous work [7, 8, 14] have successfully used SentiStrength [16] to perform sentiment analysis of software engineering interactions, such as commits, issues, and pull requests. Therefore, in this paper, we decided to use SentiStrength to perform the sentiment analysis.

SentiStrength uses a lexicon approach based on a dictionary or words and idiomatic expressions to detect two sentiment polarizations, negative (from -1 slightly negative to -5 very negative) and positive (from 1 slightly positive to 5 very positive). By that, it provides the overall sentiment of the sentence(scale), subtracting the negative sentiment from the positive sentiment [16]. We used SentiStrength to extract the sentiment of each pull request and comment.

However, Novielli et al. [10] warned that using any sentiment tool to evaluate sentiment on software engineering artifacts without any change might result in an inadequate analysis because some terms considered as negative in other social networks are natural when analyzing technical texts. For example, the word 'static,' which in the SentiStrength default dictionary is considered as -2 negative, is used most of the time by developers to reference a method or field of a class. Additionally 'static' is a common term in some programming languages such as Java. Based on the Novielli et al. [10] suggestion, and in the previous work [14, 8], we decided to change the default SentiStrength dictionary to evaluate our data set better.

Sentiment Analysis in Software Engineering: Using any sentiment analysis tool without any change will, as we had introduced, result in an inadequate analysis. For that matter, we decided to change the default SentiStrenth dictionary to address the software developer field better. We performed this process by checking the results of the classification manually and removing the common terms from the dictionary, classified as negative or positive.The Table 1 shows the modifications we have made in the dictionary; the words in the first column, *Words*, are the words that we have modified; the second column, *Original Value*, shows the value that the words have in the original SentiStrength dictionary; the third column shows the new values that we had set to the words, and the last column shows a short explanation why we had changed that group of words.

Words	Original Value	Change	Reason
broke*, fail	-2	0	Usually is a reference to the build status and has no sentiment
bug, defect, error, missing, mock,	-2	0	No sentiment related just reference fact
constrain*, drop, kill, static	-2	0	Common term in development with no sentiment expressed
Default, exit	-2	0	A common term in development with no sentiment expressed
garbage, vagrant, storm	-3	0	A common term in java projects with no sentiment expressed
revert	-2	0	Common term across GitHub social network
not working	-4	0	Idiomatic expression that most of the times have no sentiment expressed

Table 1: SentiStrength dictionary changes

Mood Variation: To evaluate the influences of others in the developer mood, we decided to use a metric called subjective well-being initially presented by Bollen et al. [3] and used in the Twitter social network to measure mood propagation. We used the technique to calculate the state of developer sentiment by analyzing a window of time. The subjective well-being ($\mathcal{S}(d)$) of a developer is given by subtracting the number of positive comments from the number of negative comments, divided by the number of positive comments plus the number of negative comments. This way, the $\mathcal{S}(d)$ value variates from -1 to 1; Equation 1 shows the $\mathcal{S}(d)$ equation. Once $\mathcal{S}(d)$ gives the developer sentiment on a specific window of time, we use the metric to evaluate the sentiment change by calculating the difference between before ($\mathcal{S}_b(d)$) and after ($\mathcal{S}_a(d)$) another developer comment on a pull request. The metric of mood variation($\mathcal{SC}(d)$) variates between -2 and 2, and Equation 2 presents its equation. As previously mentioned, we *cannot* establish a strong causal relationship between a comment and the developer mood variation, given the possibility of external influences. However, by analyzing different windows of time (1, 2, 4, 8 hours and one day) we intend to reduce or mitigate such a problem.

$$\mathcal{S}(d) = \frac{N_p(d) - N_n(d)}{N_p(d) + N_n(d)} \quad (1)$$

$$\mathcal{SC}(d) = S_b(d) - S_a(d) \quad (2)$$

3 Experiment Design

This section presents the steps taken in our experiment, starting with a short presentation of our research questions, followed by data set and finally the sentiment mining.

Goal and research questions: As we previously mentioned the primary goal of this paper is to analyze the impact of feedback on GitHub in developers mood and how the influence behaves in time. Therefore, we formulate the following research questions (RQ):

RQ1: Can a developer change the sentiment of another developer with a pull request comment?

RQ2: Does the role of the developer in the project change the intensity of influence he/she has on the sentiment of another developer?

RQ3: How does the influence behave over time? Does the behavior change depending on the comment sentiment?

RQ4: Do negative comments on the first pull request lead to quitting the project?

Data Set: We collected the data from the GitHub API. We first obtained the most popular java projects from the API [2]. GitHub limits the search to the first 1,000 results. To get the most popular projects, we sorted the results by stars. We decided to remove all the projects with less than 1,000 lines of java code because they were probably documentation or experimental projects. After filtering out projects smaller than 1,000 lines of code, 930 projects remained on the data set. After filtering the projects, we collected the project interactions, pull requests, pull request comments, pull request reviews and commits from the GitHub API. Then we filtered the 100 most popular projects among the 930 filtered in the first filter to investigate more deeply. At the end of the data collection, our data set contained 100 projects, 555,665 commits, 78,475 pull requests, 226,446 reviews, 240,060 pull comments and 15,865 developers.

Sentiment mining: After collecting the data, we applied the SentiStrenth [16] with the changes in the dictionary we presented previously, in all the interactions. We noted that the developers express less sentiment on the commits interactions, (15.52% of the commits have some sentiment expressed), on the other hand, 54.32% of the pull comments express some emotion Figure 2 shows the percentage for all the kinds of interactions. In all the cases the rate of positive sentiment is more significant than the rate of negative, on pull comments 40.5% are positive against 13.82% negative, and in the commits, the difference is small (7.86% positive,

[2]https://developer.github.com/v3/projects/

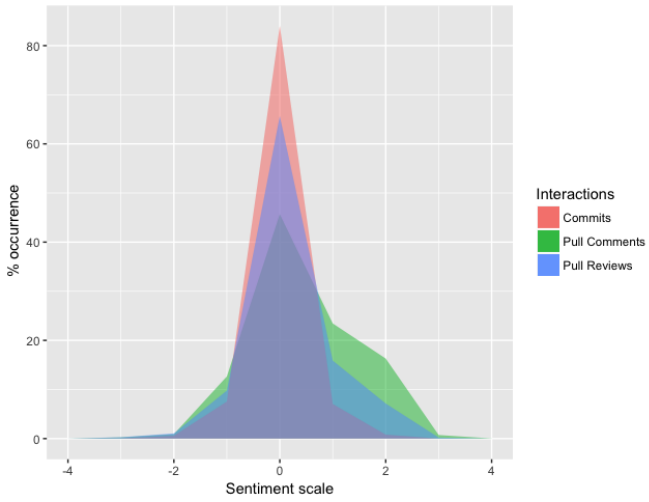

Figure 2: Sentiment distributions in different interactions.

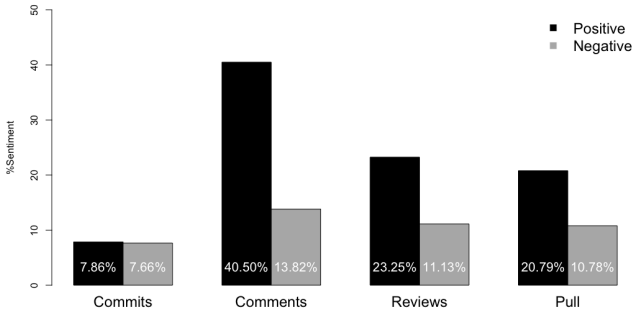

Figure 3: Sentiment by interaction.

and 7.66% negative). Figure 3 shows the percentage for all the interactions.

We noted that the most sentimental interaction is the pulls comments because this is where the users express their sentiments the most (54.32%). We expected this result, given the nature of the interaction once it represents a commentary from a developer on another developers patch of code. The commenter can agree or disagree with the change and can also request changes which sometimes starts a discussion. We believe this interaction can be a trigger for changing the humor of the developer submitting the code to integrate and we explore this hypothesis in the section 4.

4 Results

This section presents the results of the experiment each result is related to an RQ.

Mood Variation : To answer the RQ1 and RQ2, we analyzed the subjective well-being change that was previously present in the equation, for 1, 2, 4, 8 hours and one day, using a comment in a developer's pull request as a reference; this way, we can evaluate whether or not the comment

Role	%
ANY	31.16%
COLLABORATOR	30.51%
CONTRIBUTOR	31.50%
MEMBER	32.06%
NONE	29.77%
OWNER	31.62%

Table 2: Relevant sentiment change One hour time window

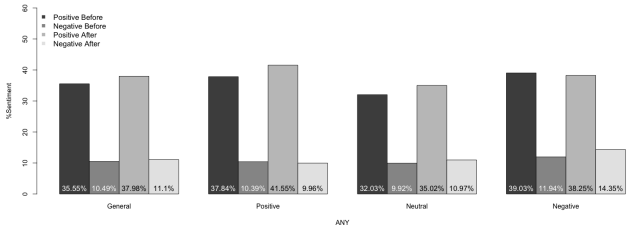

Figure 4: Sentiment variation before and after interaction with ANY developer, in the one-hour time window.

had an influence and how the influence will behave in time. Next, we discussed the results.

In a 1 hours time window, we noted 31.15% of significant mood change ($abs(\mathcal{SC}(d)) > 1$) in general. We also explored this sentiment change with the role of the commenter, but the role of the commenter does not change the influence significantly, where the smaller influence is from the role none (commenter no association with the repository) 29.77% and the highest is from member (commenter is a member of the organization that owns the repository) 32.06%. Table 2 shows the values for all the roles. We also analyzed how the sentiment changed based on the sentiment expressed by the commenter. This time, we analysed the positive and negative sentiments expressed before and after a comment. Figure 4 shows that receiving a positive comment the increased the percentage of positive interaction from 37.85% to 41.55% and receiving a negative comment increased the percentage of negative interaction from 11.94% to 14.35% ignoring the role of the commenter.

To answer **RQ3**, we studied how the influences behaved over time. We noted that as time passed, the notable possible influence reduced, as we show in Figure 5. The blue line shows the relevant mood variation ($abs(\mathcal{SC}(d)) > 1$) for each time window when receiving a neutral comment, the red shows the same when receiving negative comments, and the green line when receiving positive comments. We found that as time passed, the influence reduced in all the cases. The negative comments had the bigger influence in all analyzed time windows, with an average of 2,93 percentage points bigger than general.

To address **RQ4**, we explored the consequences of a negative comment on the first pull request of a developer.

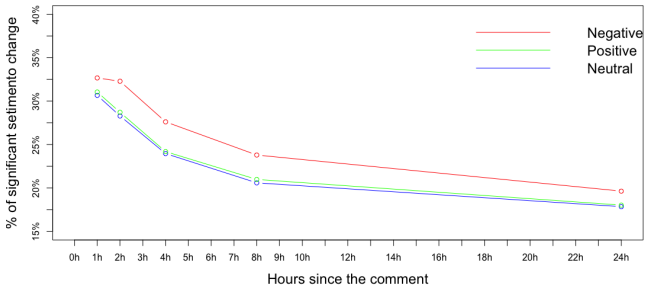

Figure 5: Relevant sentiment change by time since comment.

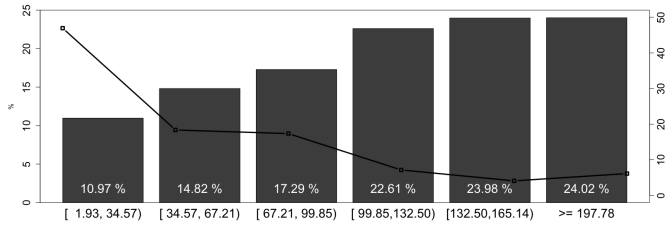

Figure 6: Percent of Once Contributor, with negative comment on the first Pull Request by KLoc

On average, 70% of the pull requests came from a contributor who would never contribute again to the repository (once-contributors). We related this to receiving a negative comment on the first contribution (pull request) and also to the project characteristics, KLoc, number of contributors, and stargazers. We found that as the project grew in KLoc(Thousand lines of code) the number on once-contributors with negative comments on the first pull request grew, from 10.97% on projects with less than 30.57 KLoc to 24.02% on projects bigger than 197.78 KLoc, we present the progression in Figure 6 in the bars, and the line shows the percent of projects in the respective KLoc range, we did not find a relationship between number of contributors or stargazers with the percentage of once-contributors.

Research Questions: In this paper, we addressed the following research questions:

RQ1: Can a developer change the sentiment of another developer with a pull request comment? The selected metric presented 31.16% of significant sentiment change when considering one hour before and after a comment.

RQ2: Does the role of the commenter in the project change the intensity of influence he/she has on the sentiment of another developer?

The role of the commenter had not had a significant impact on the sentiment influence, with a difference of only 1.83 percentage points between the most influencer when the commenter was the owner of the project and the least influencer when the commenter has no relationship with the project.

RQ3: How does the influence behave over time? Does the behavior change depending on the comment sentiment?

In general, the influence reduces as the time pass, from 31.15% in one hour to 18.16% in one day. The behavior does not change, depending on the sentiment of the comment, but we noted the biggest influence from negative comments in all the analyzed time windows.

RQ4: Do negative comments on the first Pull Request lead to quitting the project?

Only 14.85% of the developers with a single Pull request received at least one negative comment on the pull request. On the other hand, we found a weak correlation with the size of the project in lines of code and the number of developer with only one contribution that received at least one negative comment, where the percentage grew from 10.97% on projects with less then 34.57k lines of code, to 24.02% on projects bigger than 197.78k lines of code.

5 Related Work

This section describes previous works regarding sentiment analysis on software development interactions and a short comparison with the current paper.

Robinson et al. [12] performed data analysis on open source projects looking to understand how behavior change can change developers sentiment, and they analyzed 2 points, behavior change and routine change and their relationship with sentiment change. They used a regression model to search for the relations ship between developer sentiment change and developer behavior change. They hypothesized that routine change would change developer sentiment positively or negatively. They evaluated 124 GitHub projects also performing intra-project and multi-project analysis and their results shown that routine change had a positive impact on the developer sentiment when evaluation multi-project approach and negative sentiment change were related to behavior change.

Islam and Zibran [8] performed an analysis of 50 projects with more than 490 thousand commits messages, searching for sentiment variations over the commits messages. They searched for a relationship between, weekday, day hour, commits message length, and task type related to the developer sentiment variation, they used hierarchical algorithm clustering to perform clustering and used a statistical approach to support their findings. They found relationships between the task type and the developer sentiment variation; bug fix commits have more positive sentiment than refactoring tasks. Also more significant commits message express more sentiment than small commit messages, weekdays and hours of that day did not show any significant relationship with developer sentiment. Souza and

Silva [14] studied the sentiment related to building status, evaluated 1,262 projects from GitHub, and more than 609k builds, they found that the commit message following broken builds has a week correlation with negative sentiment.

Sinha et al. [13] analyzed the sentiments of the developer in the commits messages; their research focuses on finding the sentiments variations only into commits and relate this with the day of the week and the size of the commit. They found that a low percentage of commits has sentiment expressed, and there are more negative than positive sentiment expressed (5% positive and 14% negative), they also found the worst day in sentiment level (higher volume of negative sentiment) on Tuesday.

None of the related works explored the sentiment contagion studying the influence of other developers in the developers' mood; also, few papers analyzed the sentiment over the Pull Request comments, which is the proper place to promote discussion related to code. Therefore, in this paper, we take the challenge of investigating how other developers' comments may influence the developers' mood.

6 Conclusions and Future Work

This paper analyzed the impact of feedback on developers' mood when submitting pull requests. Our results showed that developers feedback might influence another developers' mood; negative comments have bigger impacts on mood variation; and as project grows in lines of code, the bigger is the impact of negative comments on the first contribution, and it might result in not contributing again with the same repository. We believe our results will help project leaders and companies create conduct codes to guide developer feedback constructively.

In future works, we intend to explore deeper the consequences of politeness and impoliteness in the success of open source projects and communities' growth or decline and its relationship with the maintainer's sentiment expression. We also intend to use the relevance of developers in a community instead of the rule they have in the project they are commenting or contributing. We believe that developer relevance has a relationship with the impact it can cause, independent of the rule they have in the project they are contributing.

Acknowledgments

This research was partially supported by CAPES, CNPq, FAPEMIG, and FIP-PUC Minas

References

[1] Comment author association — github developer guide. URL https://developer.github.com/v4/reference/enum/commentauthorassociation/.

[2] Github glossary - user documentation. URL https://help.github.com/articles/github-glossary/.

[3] Johan Bollen, Bruno Gonçalves, Guangchen Ruan, and Huina Mao. Happiness is assortative in online social networks. *CoRR*, abs/1103.0784, 2011.

[4] Munmun De Choudhury and Scott Counts. Understanding affect in the workplace via social media. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 303–316, 2013. ISBN 978-1-4503-1331-5.

[5] A. Fountaine and B. Sharif. Emotional awareness in software development: Theory and measurement. In *2017 IEEE/ACM 2nd International Workshop on Emotion Awareness in Software Engineering (SEmotion)*, pages 28–31, 2017.

[6] Daniel Graziotin, Xiaofeng Wang, and Pekka Abrahamsson. Happy software developers solve problems better: psychological measurements in empirical software engineering. *PeerJ*, 2:e289, March 2014. ISSN 2167-8359.

[7] Emitza Guzman and Bernd Bruegge. Towards emotional awareness in software development teams. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2013, pages 671–674, 2013. ISBN 978-1-4503-2237-9.

[8] M. R. Islam and M. F. Zibran. Towards understanding and exploiting developers' emotional variations in software engineering. In *2016 IEEE 14th International Conference on Software Engineering Research, Management and Applications (SERA)*, pages 185–192, 2016.

[9] Mika Mäntylä, Bram Adams, Giuseppe Destefanis, Daniel Graziotin, and Marco Ortu. Mining valence, arousal, and dominance: Possibilities for detecting burnout and productivity? In *Proceedings of the 13th International Conference on Mining Software Repositories*, MSR '16, pages 247–258, 2016. ISBN 978-1-4503-4186-8.

[10] Nicole Novielli, Fabio Calefato, and Filippo Lanubile. The challenges of sentiment detection in the social programmer ecosystem. In *Proc. of the 7th International Workshop on Social Software Engineering*, SSE 2015, pages 33–40, 2015. ISBN 978-1-4503-3818-9.

[11] Filipe Nunes Ribeiro, Matheus Araújo, Pollyanna Gonçalves, Fabrício Benevenuto, and Marcos André Gonçalves. A benchmark comparison of state-of-the-practice sentiment analysis methods. *CoRR*, abs/1512.01818, 2015.

[12] W. N. Robinson, T. Deng, and Z. Qi. Developer behavior and sentiment from data mining open source repositories. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pages 3729–3738, 2016.

[13] V. Sinha, A. Lazar, and B. Sharif. Analyzing developer sentiment in commit logs. In *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, pages 520–523, May 2016. doi: 10.1109/MSR.2016.069.

[14] R. Souza and B. Silva. Sentiment analysis of travis ci builds. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, pages 459–462, 2017.

[15] Robert S. HuckmanBradley Staats. The hidden benefits of keeping teams intact, Aug 2014. URL https://hbr.org/2013/12/the-hidden-benefits-of-keeping-teams-intact.

[16] Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. Sentiment in short strength detection informal text. *J. Am. Soc. Inf. Sci. Technol.*, 61(12):2544–2558, December 2010. ISSN 1532-2882.

[17] Valley Voices. Developers don't care how much you pay them, Feb 2017. URL https://www.forbes.com/sites/valleyvoices/2017/02/09/developers-dont-care-how-much-you-pay-them/.