Text Analysis of GitHub Pull Requests of Open Source Projects Good for New Contributors

Steve Cheruiyot[1]

[1] Harrisburg University of Science and Technology

Author Note

Analytics Department

Correspondence concerning this article should be addressed to Steve Cheruiyot, .
E-mail: SCheruiyot@my.harrisburgu.edu

Abstract

Contributing to open source projects is a way for developers to build their reputation and grow their career. Previous studies have explored sentiment analysis and text similarity in analyzing GitHub projects. However, these studies have aimed at creating tools for text analysis of software engineering texts or performing generalized comparisons of projects rather than projects that fit a specific category, such as those with a reputation for being welcoming to beginner contributors. In this paper I explore the use of the two text analysis techniques as a means of identifying GitHub projects that are good for new and beginner developers. I investigate if there are shared similarities among these projects that could be used in identifying other projects that fit the category. To this end, I extract pull request descriptions and comments from a curated list of GitHub repositories with a reputation of being good for beginner contributors. I then extract sentiment polarity in the texts and compute text similarity scores between the repositories. Results show a high similarity of pull request descriptions and comments between repositories. On the other hand, sentiment analysis results differ from one repository to another. These results suggest that similarity analysis might be a more suitable approach for identifying other repositories that might also be good for beginner contributors.

Keywords: Text Analysis, Sentiment Analysis, Similarity, Open Source Software

Word count: 4,307

Text Analysis of GitHub Pull Requests of Open Source Projects Good for New Contributors

Open source projects attract contributions from developers who are motivated to contribute by a variety of reasons. Among the motivations identified by previous research are development of skills, possibility of benefiting financially from the projects and gaining reputation (Lakhani et al., 2004). Additionally, contribution to open source software projects (OSS) hosted on software development platforms such as GitHub is seen by contributors as a way of growing one's career or deriving enjoyment. As such, contributing to OSS might be an attractive endeavor for beginner contributors who are seeking to improve their skills, grow their careers or derive intellectual stimulation from writing open source software. Beginner contributors can be developers who are new to the field of software engineering or experienced developers who have not made any contributions to OSS. For these developers looking to get started on open source contribution, finding a project that is suitable to their preferences and also welcoming to new contributors would require perusing GitHub repositories, which is a time consuming task. Fortunately, there have been efforts to maintain a curated list of projects that have a history of being welcoming to new contributors (GitHub, 2018). However, since such lists depend on voluntary reporting by current contributors, they might not be exhaustive, possibly omitting projects that would otherwise be good for new contributors.

An automated approach to evaluating whether a given open source project is suitable for beginner contributors would not only help potential contributors evaluate projects faster, but also allow them to explore a larger number of projects. Consequently, this paper explores two text analysis techniques as ways of characterizing open source projects that have a history of being good for beginner contributors. The characterization would then be used to automatically identify other projects that might also be welcoming to beginner contributors. The text analysis techniques explored are sentiment and similarity analysis. These two techniques have widely been explored within the context of development of open

source software have widely been explored in previous research. For example, Zhang et al. (2017) use similarity to build a project recommendation system and Calefato et al., 2018 develop Senti4SD, a tool for extracting sentiments from software engineering texts. However, the two text analysis approaches have yet to be applied to identifying projects that are good for beginner contributors, a gap that this paper aims to fill. For instance, sentiment analysis research that has been done on open source projects hosted on GitHub has mainly focused on sentiment classification of issues, pull request comments or commit logs. The aim of the classification has been to uncover relationships between sentiment expressed and other with other variables involved such as day of the week a commit was made or geographical location of the developer who made a comment.

Literature Review

Contributing to Open Source

Research has explored open source software development extensively. One important research question has been, what motivates individuals to spend their time contributing to open source software, mostly without compensation. Lakhani and Wolf (2003) studied the motivations of individuals to contribute to open source projects using a survey administered to developers in several open source projects. The researchers find the top motivators to be how creative an individual feels when contributing, intellectual stimulation from coding, and improvement of programming skills. Other motivations include a direct need for the software being developed (either for work purposes or personal use), reputation building and improvement of professional status. Hertel et al. (2003) conducted a study of motivations of 141 contributors to the Linux kernel–a large and well-known open source project. They uncover a number of motivations, some which are specific to the Linux project and others such as career growth and enjoyment of programming support findings by Lakhani and Wolf (2003). A study based on a survey of 300 contributors to two open source projects by Oreg et al. (2008) also supports the findings of earlier research, emphasizing reputation-gaining

and skill development as motivators for contributing to open source projects.

Text Similarity

Text similarity is a natural language processing technique used to measure how close two documents or groups of text are. In the context of open source development, text similarity has been studied in previous research as a way of comparing different facets of GitHub repositories of OSS. Zhang et al. (2017) used text similarity of readme files to develop RepoPal, a recommendation system aimed at detecting GitHub repositories that are similar. Their results show that RepoPal outperforms previously developed tools such as CLAN (McMillan et al. 2012). Ma et al. (2017) investigated whether there is a high similarity between competing pull requests, that is, pull requests that make changes to the same code. Their results suggest that there exists such similarity. Li et al. (2017) used similarity of pull request titles and descriptions to detect duplicate pull requests. Their results indicate that up to 71% of duplicates can be found using text similarity.

Sentiment Analysis

Sentiment analysis, also known as opinion mining, is a widely studied field with about 7,000 research papers having published on the topic(Mäntylä et al., 2018). While most of the analysis has been done on online product reviews and social media texts, such from Facebook and Twitter, it has also been applied in the Software Engineering domain. In collaborative software engineering environments such as that of open source projects, developers express sentiments in the text they write (Jurado & Rodriguez, 2015). In the recent years, several sentiment analysis related research studies have been performed on data from open source projects hosted on GitHub, an online software collaboration tool.

Sentiment Analysis in Software Engineering

In recent years, there is a growth in research that uses sentiment analysis on software engineering texts. Notable applications include technical Q&A platforms like Stack Overflow

and collaborative development tools like Jira and GitHub. Earlier on, most of the studies reused off-the-shelf tools like SentiStrength and NLTK. However, these tools have been trained on nontechnical texts such as online product reviews, movie reviews and social media data. In their study, Jongeling et al. (2015), analyzed the performance of the off-shelf sentiment analysis tools, specifically, how they compare to human ratings and to other tools. They compare four widely used sentiment analysis tools; SentiStrength, Alchemy, NLTK and Stanford NLP. After looking at the performance of the tools, they observe that not only do the tools considered not agree with the manual labeling, but also, they do not agree with each other can result in contradictory results when used in study of software engineering texts.

Due to the unreliability of the off-the-shelf tools, there have been some research aimed at developing sentiment analysis tools customized for the software engineering domain. Software engineering specific tools currently available include Senti4SD (Calefato et al., 2018), SentiStrengthSE (Islam & Zibran, 2017), and SentiCR (Ahmed et al., 2017). Senti4SD and SentiCR are supervised classifiers while SentiStrengthSE is unsupervised classifier. A study by Novielli et al. (2018) performed a comparative study of these software engineering specific tools. The researchers used texts from Jira, Stack Overflow, Code Reviews, and Java Libraries datasets. The texts had been annotated as positive, negative or neutral using model-driven and ad-hoc annotation approaches. In the model-driven approach, annotation was based on a framework by Shaver et al. (1987) that classifies emotions as an hierarchical tree structure while in the ad-hoc approach human raters gave their subjective polarity classification of the texts. The researchers then split the dataset into training set (70%) and testing sets (30%). They then measured performance of the tools in terms of recall, precision and F-Measure scores, while using SentiStrength (an off-the-shelf tool) as a baseline for comparison. The results of the study indicate that customizing tools to software engineering texts may improve the accuracy of sentiment analysis. Unlike the off-the-shelf tools in the earlier study by Jongeling et al. (2015), the Software Engineering tools not only show good

agreement with respect to manually labeled data but also show agreement with each other.

Sentiment Analysis of GitHub Commits, Issues and Pull Requests

There have been a number of research studies on GitHub data. Areas of research have ranged from GitHub project commits (commit log), pull request and pull request comments and issues. Jurado et al., (2015) performed a general study to check the viability of using sentiment analysis on GitHub data. They conducted and exploratory case study involving an analysis of 10,829 issues from nine well-known projects on GitHub. Results show that the developers leave underlying sentiments in the text, and that the information could be used to analyze the development process. Guzman et al., (2014) performed a study of emotions (sentiment analysis) of commit comments of 90 open source software projects hosted on GitHub. The researchers were interested in finding out whether emotions expressed in the commits were related to day of the week a commit was made, team geographical distribution or project approval. They used SentiStrength, a sentiment analysis tool, for analyzing the sentiment of commit messages. The authors find that the average emotion score of commit comments for each of the projects studied tends to be neutral, commits are more negative on Monday than on Sunday, Tuesday and Wednesday, and no correlation between project approval and emotions in commit messages. The study used SentiStrength, an off-the-shelf sentiment analysis tool that has been shown to have lower accuracy compared to software engineering specific tools (Novielli et al., 2018).

Sinha et al., (2016) performed a similar study. However, they only analyzed commit messages of GitHub projects. They grouped the commits based on the day of the week they were made then used SentiStrength to determine the polarity (negative, neutral or positive) of the commit messages. Results of the study indicate that majority of commits' sentiment is neutral, which might be attributed to the use of an off-the-shelf tool (SentiStrength), whose shortcomings have led to the development software engineering specific tools discussed earlier. Huq et al., (2019) studied how the sentiment of developers relates to bugs in code.

They extracted sentiments from pull requests of well-known GitHub repositories. The authors then conducted sentiment analysis on the pull requests using SentiStrength-SE, a tool customized for analyzing software engineering texts. They then analyzed the difference in pull request comments' sentiment between regular code changes and those changes that accidentally introduce bugs, also known as Fix Inducing Changes (FIC). Results of the study show that pull request comments are more negative for FICs than for regular changes.

Purpose Statement and Research Questions

The purpose of this paper is twofold. First, to investigate the overall sentiment of pull requests descriptions and comments and compare how similar they are across a set of open source projects hosted on GitHub that have been identified as being good for new contributors. Second, to examine if there are any shared characteristics among the projects which could be used as a way of identifying other projects that might be welcoming to new contributors.

Research Questions

RQ1: What is the underlying sentiment in pull request descriptions and comments of projects that are good for new contributors? This question aims to identify the sentiment expressed in pull request descriptions and comments.

RQ2: How similar are pull request descriptions and comments of projects that are good for new contributors? This question aims to compare pull request descriptions and comments to establish how similar they are semantically.

RQ3: How does sentiment analysis and similarity analysis compare across projects that are good for new contributors? This question will be concerned with comparing the results of sentiment analysis with that of similarity, with the aim of uncovering any shared patterns across the projects.

Methods

This section covers the methodology used in this paper. To answer the three research questions outlined earlier, a quantitative analysis of results derived from analyzing the text of pull request (PR) descriptions and comments was conducted. The PR descriptions and comments extracted from 14 repositories identified as being good for beginner contributors. For the first research question, I analyzed an aggregation of sentiments expressed in PR descriptions and comments. To answer the second question, I measured the similarity of PR descriptions and comments between repositories. Lastly, I compared the results from the first and second analyses by means of correlation coefficient and tested the outcomes for significance.

Data

Pull request descriptions and comments data was retrieved from source repositories using GitHub's public API. The repositories included *nodejs/node*, *moby/moby*, *django/django*, *atom/atom*, *rust-lang/rust*, *hoodiehq/hoodie*, *OperationCode/START_HERE*, *zulip/zulip*, *middleman/middleman*, *Homebrew/brew*, *beeware/batavia*, *exercism/exercism*, *howdyai/botkit*, *HospitalRun/hospitalrun-frontend* (the format used here is *owner/repository* where an owner can be a GitHub user or an organization). These repositories were selected as they have a historical reputation of being welcoming to beginner contributors according a GitHub showcase (GitHub, 2018). The 14 repositories make up the entire showcase list. The initial goal was to choose repositories from based on programming language used in a repository. However, due to diversity in programming languages used (notably *JavaScript*, *TypeScript*, *Go*, *Rust* and *Ruby*), picking any one language would only reduce the diversity of the dataset.

Retrieving data from GitHub's API was a two-step process. First, for each repository, pull request descriptions were downloaded in JSON format, for a maximum of 100 pull requests per repository. Second, for each extracted pull request, accompanying pull request

comments were downloaded in JSON format as well, again with a maximum of 100 comments per pull requests. The raw JSON data retrieved from the GitHub API included pull requests and comments metadata such as the `user` who created the resource, pull request `number`, URLs to GitHub's website location of the pull request, when the pull request or comment was created and links and references relevant to the PR or comment. Since this project is on text analysis, the variables of interest from the raw JSON were `title`, title of a PR, and `body`, PR description text or comment text. Table 1 and Table 2 provide a summary the data extracted from GitHub.

Variables

Since this project is on text analysis, the variables of interest were pull request descriptions and pull request comments. Both of these variables were labeled as `body` in the extracted data. Table 1 below shows a summary of the length of pull request descriptions in each repository, for instance, we see that the repository *Homebrew/brew* has the longest pull request descriptions, 2,575 characters on average. The table also shows a truncated example of a random pull request description. Table 2 on the other hand shows the total number of pull request comments for each repository and a truncated example of a pull request comment.

Table 1

Overview of unprocessed pull requests data showing the number of PRs for each repository and summary statistics for the length PR descriptions. An example of a PR description is included as well.

Repository	No of PRs	Mean	Median	Min	Max	Example
atom	100	1,353.92	999.00	0	13929	NA
batavia	2	895.50	895.50	782	1009	<!— Describe your changes in detail –> <!—...
botkit	8	740.75	277.00	0	3807	
brew	21	2,575.14	1,317.00	56	19349	- [x] Have you followed the guidelines in our [...
django	100	307.02	151.50	0	2260	It always helps to have an example.
exercism	1	145.00	145.00	145	145	I know that you might not need an spanish versi...
hoodie	23	1,318.39	103.00	0	6082	a minor change in readme.md some grammatical ch...
hospitalrun-frontend	12	386.92	294.50	112	776	Closes #2364 Allows notes on labs to be delete...
middleman	13	918.62	466.00	29	4119	Bumps [rubocop-performance](https://github.com/...
moby	100	1,150.56	915.50	10	5384	**- What I did** Added Neil Armstrong to the n...
node	100	1,880.17	1,823.50	23	8022	NA
rust	100	461.17	204.00	0	6436	Some HIR nodes are guaranteed to be HIR owners:...
START_HERE	5	57.80	35.00	0	158	I removed the Hacktoberfest event banner and re...
zulip	100	641.86	565.50	0	2602	We currently don’t support editing widgets, so ...

Table 2

Overview of unprocessed pull request comments data showing the number of PR comments for each repository and an example of a comment.

Repository	No of comments	Example
atom	309	Note: this won't affect pasting in Atom that much: copyin...
batavia	1	Aside: I see a warning when running 'make html' which app...
botkit	11	link qr code is: line://au/q/ZNXJ2yXgETQ30BUDOGnaG4RXS1YN...
brew	139	<!– #review-period-begin –> Review period will end on ...
django	333	Thank you for looking into this.
hoodie	53	NA
hospitalrun-frontend	54	<a href="https://gitpod.io/#https://github.com/Hospital...
middleman	53	This is very interesting. Thank you. Let me research this...
moby	291	related issues: https://github.com/moby/moby/issues/37344...
node	507	CI: https://ci.nodejs.org/job/node-test-pull-request/35813/
rust	494	Minor nitpicks. Ping me when done and I'll r=me once addr...
zulip	191	@ganpa3 why did you squash the 2 commits that were doing ...

Data Analytic Plan

After retrieval, the raw data was pre-processed to prepare it for analysis. Pre-processing comprised sub-setting the raw data to only include variables of interest followed by cleaning textual data. For pull request data, variables that were retained included `owner` and `repository` for identification purposes, `id`, `title` and `body` (PR description), `state` of a PR, whether it's open or closed, PR `number` assigned by GitHub and timestamps. For PR comments, variables retained were `body` (comment's text) and `id`, with `owner` and `repository` names added to them. Cleaning textual data was applied to `body` and `title` variables, which are also the target variables on which analysis will be performed. The steps applied in text cleaning were converting the texts to lower case, normalizing unicode characters into ASCII encoding, removing URLs, symbols and white space, expanding word contractions (e.g `don't` to `do not`) , lemmatization (reducing words to their base form e.g *studying* to *study*), and finally removing stop words with exception of *no*, *not* and *but* which might be helpful in sentiment analysis. After cleaning, rows containing texts with less than 10 words were dropped to simplify analysis. Additionally, repositories with less than 10 pull requests were excluded from the final analysis. This step left 10 repositories, excluding *OperationCode/START_HERE*, *beeware/batavia*, *exercism/exercism* and *howdyai/botkit*.

The data analysis plan for this project was divided into three steps. First, sentiment polarity for both pull requests descriptions and comments was extracted, for each of the 10 repositories left after data cleaning. Sentiment analysis was performed using Senti4SD, a tool developed by Calefato et al. (2018) to support sentiment analysis of software engineering texts. The output of Senti4SD was sentiment polarity classification of a PR descriptions and comments as *negative*, *neutral* or *positive*. The second step involved measuring the cosine similarity of combined PR descriptions and comments of one repository to those of another. The final step was to compare the results of sentiment analysis with those of similarity

measures to see if there was any relationship between the two.

Senti4SD is a machine learning classifier trained on labeled data extracted from Stack Overflow. It categorizes a text as *negative*, *neutral* or *positive*. In developing the classifier, three sets of features were extracted from the dataset. First, a lexicon-based set of features using SentiStrength's lexicon. Second, a set keyword-based features which include word counts for uni-grams and bi-grams. Third, semantic features which are word embeddings generated using continuous bag-of-words (CBOW) method in Word2vec. The features were then used to train an Support Vector Machine (SVM) classifier, resulting in the model that is Senti4SD. Table 3 shows examples of original PR description texts and the polarity classes predicted by Senti4SD.

Table 3

Examples of raw PRs descriptions and their polarity classification as predicted by Senti4SD.

	Text	Predicted Polarity
90	https://github.com/nodejs/node/issues/36154 <!– Thank you for your pull request. Please provide...	positive
92	This is used to handle support for **WebAssembly** modules and static analysis of named exports f...	positive
493	‘super_relate_consts‘ eagerly evaluates constants which doesn’t seem too great. I now also final...	positive
86	Currently, those exceptions were swallowed and the execution may continue in an unstable state. ...	negative
176	Signed-off-by: Quang Kieu <quangkieu1993@gmail.com> <!– Please make sure you’ve read and unders...	negative
316	### Description of the Change This is an update of the default syntax themes to implement [namin...	negative
76	Make no-op direct calls of ‘Set‘ prototype methods to ‘process.allowedNodeEnvironmentFlags‘. “‘...	neutral
481	The derived implementation of ‘partial_cmp‘ compares matching fields one by one, stopping the com...	neutral
168	This is a long-requested feature, and it solves several usability problems with the default AppAr...	neutral

Results

In this section I present the results of the analysis conducted and address each of the research questions.

RQ1. What are the underlying sentiments in PR descriptions and comments?

This research question involved conducting sentiment analysis on pull request (PR) descriptions and comments. For each repository (GitHub project), sentiment analysis was performed in two steps, first on PR descriptions then on PR comments. Polarity, which falls into three classes, *negative*, *neutral* or *positive* was predicted for each PR description and comment using Senti4SD, a tool developed by Calefato et al. (2018). Table 4 shows a random sampling of polarity predictions for PR descriptions.

Table 4

A random sampling of polarity classification of PR descriptions as predicted by Senti4SD.

	Repository	Text	Predicted Polarity
378	rust-lang	add tidy check markdown file think useful style enforceme...	negative
382	rust-lang	long ago benchmark seemed say function performance benefi...	neutral
21	nodejs	fix submitting pull request please read commit message fo...	positive
141	moby	backport please make sure read understood contributing gu...	negative
169	moby	please make sure read understood contributing guideline m...	positive

After computing the polarity of each PR description and comment, I summed up the number of PR descriptions and comments for each of the three polarity classes then computed the percentage of documents per polarity class for each the repositories analyzed. For instance, for the repository *middleman*, 27.27% of PR comments had a *negative* polarity, 45.45% *neutral* and 27.27% a *positive* polarity. Table 5 and Table 6 summarize the distribution of polarity classes in each repository's PR descriptions and comments.

Table 5

Percentage of PR descriptions for each polarity class grouped by repository.

Repository	Negative	Neutral	Positive	Neutral+Positive
atom	24.18%	34.07%	41.76%	75.82%
django	9.43%	50.94%	39.62%	90.57%
Homebrew	0.00%	5.00%	95.00%	100.00%
hoodiehq	0.00%	27.27%	72.73%	100.00%
HospitalRun	25.00%	41.67%	33.33%	75.00%
middleman	27.27%	45.45%	27.27%	72.73%
moby	24.14%	24.14%	51.72%	75.86%
nodejs	26.32%	3.16%	70.53%	73.68%
rust-lang	30.88%	52.94%	16.18%	69.12%
zulip	5.68%	25.00%	69.32%	94.32%

From the results in Table 5 and Table 6, there is no clear pattern in the distribution of PR descriptions and comments across polarity classes. In most of the repositories, *negative* sentiment polarity accounts for the fewest number of both PR descriptions and comments. In each of the repositories, *neutral* and *positive* sentiment polarities account more than 65% of PR descriptions and PR comments.

RQ2: How similar are PR descriptions and comments?

To assess how similar PR descriptions and comments are, I computed the semantic similarity between pairs of repositories. For each of the repositories, I combined PR descriptions into a single text document and PR comments into another. Given there are 10 repositories being analyzed, the resulting number of unique repository pairs was 45. I then computed the similarity score for each pair using spaCy, an open-source software library for

Table 6

Percentage of PR comments for each polarity class grouped by repository.

Repository	Negative	Neutral	Positive	Neutral+Positive
atom	18.78%	42.25%	38.97%	81.22%
django	29.07%	27.31%	43.61%	70.93%
Homebrew	32.10%	38.27%	29.63%	67.90%
hoodiehq	6.00%	74.00%	20.00%	94.00%
HospitalRun	2.78%	55.56%	41.67%	97.22%
middleman	28.57%	32.14%	39.29%	71.43%
moby	30.77%	37.82%	31.41%	69.23%
nodejs	21.55%	38.36%	40.09%	78.45%
rust-lang	34.40%	26.80%	38.80%	65.60%
zulip	11.11%	48.72%	40.17%	88.89%

natural language processing (Honnibal et al., 2020), first for PR descriptions followed by PR comments. The similarity scores shown are cosine similarity scores which range from 0 to 1, with 0 implying two documents being compared are not similar at all and 1 implying that the documents are semantically identical.

PR descriptions have a high similarity score ($M = 0.92$ and $SD = 0.05$) with the lowest score being a 0.80. For PR comments, while the average similarity score is also high ($M = 0.87$ and $SD = 0.11$), we see a comparably lower minimum score of 0.66. Table 7 shows a listing of similarity scores for PR descriptions and comments for each of the 45 repository pairs. *django* and *atom* (0.98) have the most similar PR descriptions while *nodejs* and *hoodiehq* are the least similar PR descriptions (0.80). For PR comments, *django* and *atom* (0.99) are also the most similar whereas *rust-lang* and *HospitalRun* are the least similar (0.66)

Table 7

PR descriptions and PR comments similarity scores between repository pairs.

1st Repo	2nd Repo	Similarity Score	Score (PR comment)
atom	Homebrew	0.91	0.98
atom	hoodiehq	0.86	0.83
atom	HospitalRun	0.97	0.97
atom	middleman	0.97	0.99
atom	rust-lang	0.95	0.74
atom	zulip	0.98	0.98
django	atom	0.98	0.99
django	Homebrew	0.89	0.98
django	hoodiehq	0.83	0.83
django	HospitalRun	0.97	0.97
django	middleman	0.96	0.99
django	rust-lang	0.95	0.74
django	zulip	0.98	0.98
Homebrew	HospitalRun	0.85	0.95
hoodiehq	Homebrew	0.96	0.80
hoodiehq	HospitalRun	0.80	0.82
hoodiehq	middleman	0.89	0.81
hoodiehq	zulip	0.82	0.83
middleman	Homebrew	0.92	0.98
middleman	HospitalRun	0.94	0.97
moby	atom	0.98	0.97
moby	django	0.98	0.97
moby	Homebrew	0.89	0.96
moby	hoodiehq	0.83	0.85

Table 7 continued

1st Repo	2nd Repo	Similarity Score	Score (PR comment)
moby	HospitalRun	0.97	0.93
moby	middleman	0.96	0.95
moby	rust-lang	0.95	0.83
moby	zulip	0.98	0.96
nodejs	atom	0.96	0.83
nodejs	django	0.96	0.83
nodejs	Homebrew	0.85	0.80
nodejs	hoodiehq	0.80	0.74
nodejs	HospitalRun	0.95	0.76
nodejs	middleman	0.92	0.79
nodejs	moby	0.95	0.88
nodejs	rust-lang	0.91	0.85
nodejs	zulip	0.95	0.79
rust-lang	Homebrew	0.90	0.70
rust-lang	hoodiehq	0.86	0.72
rust-lang	HospitalRun	0.92	0.66
rust-lang	middleman	0.94	0.68
rust-lang	zulip	0.93	0.69
zulip	Homebrew	0.87	0.97
zulip	HospitalRun	0.97	0.98
zulip	middleman	0.96	0.98

RQ3: How does sentiment analysis and similarity analysis compare?

To compare the results of sentiment analysis and similarity, I computed the correlation between proportions of each polarity class (*negative*, *positive* or *neutral*) and similarity scores. To this end, the similarity score of a repository was calculated as the average of scores for repository pairs of which the repository was one of the pairs. Figure 1 displays scatter plots of the derived similarity scores and polarity proportions for PR descriptions and comments along with the Pearson correlation coefficient for each of the relationships.

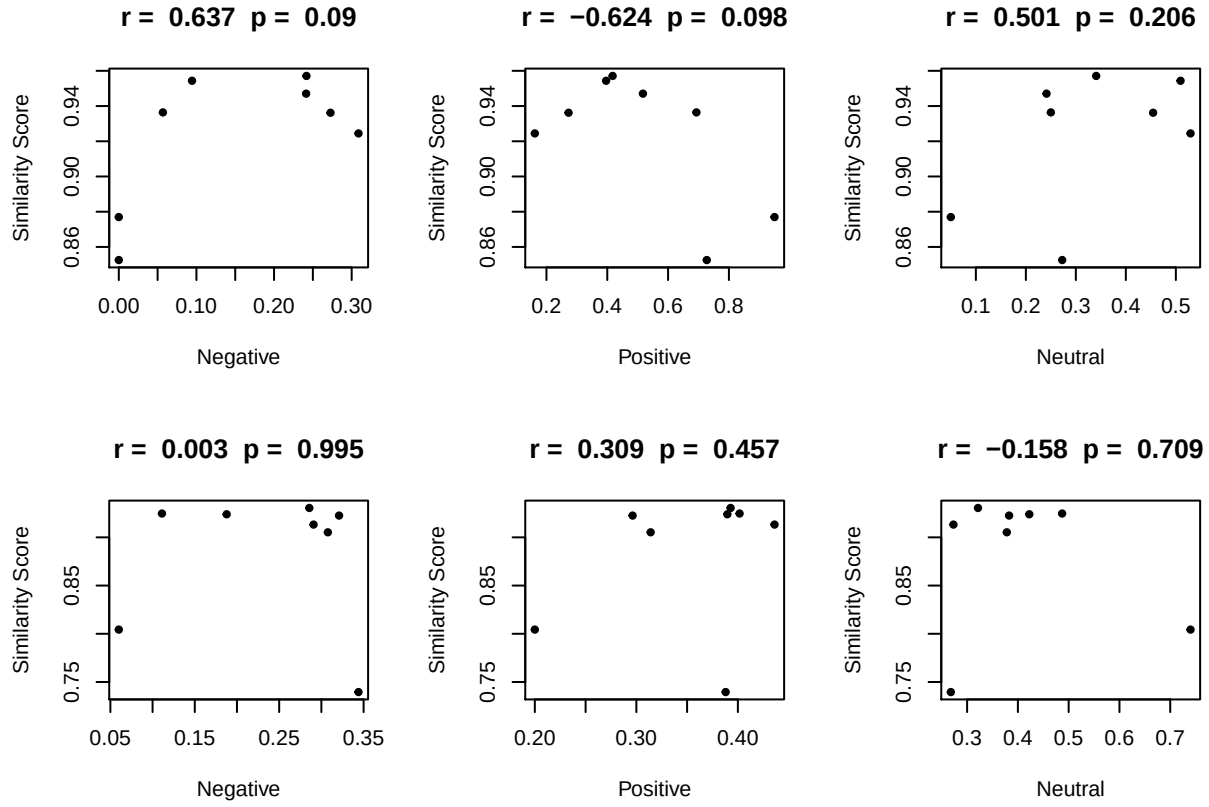

Figure 1. Scatter plots of PR descriptions (top row) and comments (bottom row) polarity proportions and average similarity score between a repository and other repositories. Pearson's r is also shown for each relationship.

Correlation results for PR descriptions indicate a moderate positive correlation between the proportion of documents with a *negative* polarity and the similarity score between a repository and other repositories ($r(6) = .637$, $p = .090$). On the other hand, we

see a moderate negative correlation for *positive* polarity ($r(6)$ = -.624, p = .098). For *neutral* polarity, the correlation is positive ($r(6)$ = .501, p = .206). For PR comments, correlations results from those of PR descriptions. There is no correlation between *negative* polarity and similarity score ($r(6)$ = .003, p = .995). For *positive* polarity we see a weak positive correlation ($r(6)$ = .309, p = .457) and finally for *neutral* polarity, a weak negative correlation ($r(6)$ = -.158, p = .709). However, for both PR descriptions and comments, none of the correlations between proportions of polarity classes and similarity scores are significant. The scatter plots in Figure 1. support these results as we see no clear linear relationships between polarity and similarity scores.

Discussion

Summary of Results

RQ1: What is the underlying sentiment in pull request descriptions and comments of projects that are good for new contributors? Sentiment analysis results show a varied distribution of proportions of PR descriptions and comments that are of negative, positive or neutral polarity from one repository to another. The lack of any observable patterns across the repositories analyzed suggests that repositories that are good for beginners might not have shared similarities with regards to sentiments expressed in their PR descriptions and comments. This outcome differs from an expected result that the repositories would similar sentiment patterns. A likely explanation for this outcome is that Senti4SD (Calefato et al., 2018), the tool used for sentiment polarity classification, was developed using data from Stack Overflow. As a result, the tool might have a lower precision when used on data from other platforms such as GitHub.

RQ2: How similar are pull request descriptions and comments of projects that are good for new contributors? Similarity measures of both combined PR descriptions and comments were consistently high between the repositories analyzed in both PR descriptions and comments. This suggests that PR descriptions and comments are highly similar across GitHub projects that are good for beginner contributors. In a related work, Li et al. (2017) find that 71% of duplicate pull requests can be found using text similarity similarity of PR titles and descriptions. While the researchers' work focused on comparing PR descriptions within a single repository, the findings in this paper demonstrate that text similarity could be applied to comparing PR descriptions of more than one repository.

RQ3: How does sentiment analysis and similarity analysis compare across projects that are good for new contributors? Correlation analysis between the results of sentiment analysis and text similarity showed no significant relationship between the findings of two text analysis techniques. Even though there was no significant correlation, the result supports the

findings in the first two research questions, i.e., we see no similarities across repositories for sentiment analysis but high similarity scores in the second technique. This suggests that the results of the two approaches can be interpreted independently.

Limitations

A limitation of this study is that curated lists of open source projects that have a history of being welcoming to new contributors is dependent on reporting by active contributors. Therefore, the lists might not include all possible projects that fit the category. Additionally, the contributors who report a project as being welcoming to beginners might not be an accurate representation of the sentiment of all the contributors in the project. A possible way to address this limitation would be to conduct a survey of contributors to a random sample of open source projects hosted on GitHub. Candidate projects for undergoing sampling might be selected based on criteria such as the number of contributors, programming language or popularity (based on number of stars on a repository).

Future Directions

The next steps for the topic presented here would be conduct similar analysis on GitHub projects with a reputation of being unwelcoming to new and beginner contributors. This would help further assess whether similarity would be a suitable approach for establishing projects that are similar to the ones used in analysis are also good for new and beginner contributors. In addition to analyzing additional projects, future work could incorporate additional documents from GitHub projects such as documentation files (e.g. Readme.md) on top of PR descriptions and comments.

Implications

The work presented here illustrates two approaches to analyzing PR descriptions and comments of GitHub projects that are good for new and beginner contributors. Results show that repositories known to be good for beginner contributors have a strong textual similarity

of their PR descriptions and comments. This is a useful finding as it might be used as a basis for developing automated tools that evaluate whether a given repository might be good for a beginner contributor. This would be achieved by comparing the similarity of PR comments and descriptions of a candidate repository against those of the projects already known to be good for beginners. Such tools would from would potentially save the time spent exploring projects to contribute to by beginner contributors. Moreover, the results of the study emphasizes text similarity as a technique for comparing different aspects of open source software development on GitHub.

Acknowledgments

I would like to thank Dr. Kayla Jordan for providing feedback and guidance throughout this research.

References

262588213843476. (2018). Great for new contributors. *GitHub*. Retrieved from https://github.com/showcases/great-for-new-contributors

Ahmed, T., Bosu, A., Iqbal, A., & Rahimi, S. (2017). SentiCR: A customized sentiment analysis tool for code review interactions. In *2017 32nd ieee/acm international conference on automated software engineering (ase)* (pp. 106–111). IEEE.

Aust, F., & Barth, M. (2020). *papaja: Create APA manuscripts with R Markdown*. Retrieved from https://github.com/crsh/papaja

Calefato, F., Lanubile, F., Maiorano, F., & Novielli, N. (2018). Sentiment polarity detection for software development. *Empirical Software Engineering*, *23*(3), 1352–1382.

Charrad, M., Ghazzali, N., Boiteau, V., & Niknafs, A. (2014). NbClust: An R package for determining the relevant number of clusters in a data set. *Journal of Statistical Software*, *61*(6), 1–36. Retrieved from http://www.jstatsoft.org/v61/i06/

Cosentino, V., Izquierdo, J. L. C., & Cabot, J. (2017). A systematic mapping study of software development with github. *IEEE Access*, *5*, 7173–7192.

De Choudhury, M., & Counts, S. (2013). Understanding affect in the workplace via social media. In *Proceedings of the 2013 conference on computer supported cooperative work* (pp. 303–316).

Ding, J., Sun, H., Wang, X., & Liu, X. (2018). Entity-level sentiment analysis of issue comments. In *Proceedings of the 3rd international workshop on emotion awareness in software engineering* (pp. 7–13).

Gousios, G., & Spinellis, D. (2017). Mining software engineering data from github. In *2017 ieee/acm 39th international conference on software engineering companion (icse-c)*

(pp. 501–502). IEEE.

Guzman, E., Azócar, D., & Li, Y. (2014). Sentiment analysis of commit comments in github: An empirical study. In *Proceedings of the 11th working conference on mining software repositories* (pp. 352–355).

Hertel, G., Niedner, S., & Herrmann, S. (2003). Motivation of software developers in open source projects: An internet-based survey of contributors to the linux kernel. *Research Policy*, *32*(7), 1159–1177.

Honnibal, M., Montani, I., Van Landeghem, S., & Boyd, A. (2020). *spaCy: Industrial-strength Natural Language Processing in Python*. Zenodo. https://doi.org/10.5281/zenodo.1212303

Huq, S. F., Sadiq, A. Z., & Sakib, K. (2019). Understanding the effect of developer sentiment on fix-inducing changes: An exploratory study on github pull requests. In *2019 26th asia-pacific software engineering conference (apsec)* (pp. 514–521). IEEE.

Islam, M. R., & Zibran, M. F. (2017). Leveraging automated sentiment analysis in software engineering. In *2017 ieee/acm 14th international conference on mining software repositories (msr)* (pp. 203–214). IEEE.

Jongeling, R., Datta, S., & Serebrenik, A. (2015). Choosing your weapons: On sentiment analysis tools for software engineering research. In *2015 ieee international conference on software maintenance and evolution (icsme)* (pp. 531–535). IEEE.

Jurado, F., & Rodriguez, P. (2015). Sentiment analysis in monitoring software development processes: An exploratory case study on github's project issues. *Journal of Systems and Software*, *104*, 82–89.

Lakhani, K. R., & Von Hippel, E. (2004). How open source software works:"Free"

user-to-user assistance. In *Produktentwicklung mit virtuellen communities* (pp. 303–339). Springer.

Lakhani, K. R., & Wolf, R. G. (2003). Why hackers do what they do: Understanding motivation and effort in free/open source software projects.

Li, Z., Yin, G., Yu, Y., Wang, T., & Wang, H. (2017). Detecting duplicate pull-requests in github. In *Proceedings of the 9th asia-pacific symposium on internetware* (pp. 1–6).

Ma, P., Xu, D., Zhang, X., & Xuan, J. (2017). Changes are similar: Measuring similarity of pull requests that change the same code in github. In *Software engineering and methodology for emerging domains* (pp. 115–128). Springer.

McMillan, C., Grechanik, M., & Poshyvanyk, D. (2012). Detecting similar software applications. In *2012 34th international conference on software engineering (icse)* (pp. 364–374). IEEE.

Neuwirth, E. (2014). *RColorBrewer: ColorBrewer palettes*. Retrieved from https://CRAN.R-project.org/package=RColorBrewer

Novielli, N., Girardi, D., & Lanubile, F. (2018). A benchmark study on sentiment analysis for software engineering research. In *2018 ieee/acm 15th international conference on mining software repositories (msr)* (pp. 364–375). IEEE.

O'Hara-Wild, M., & Hyndman, R. (2021). *Vitae: Curriculum vitae for r markdown*. Retrieved from https://CRAN.R-project.org/package=vitae

Oreg, S., & Nov, O. (2008). Exploring motivations for contributing to open source initiatives: The roles of contribution context and personal values. *Computers in Human Behavior*, *24*(5), 2055–2073.

Pang, B., & Lee, L. (2009). Opinion mining and sentiment analysis. *Comput. Linguist*,

35(2), 311–312.

R Core Team. (2020). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from https://www.R-project.org/

Shaver, P., Schwartz, J., Kirson, D., & O'connor, C. (1987). Emotion knowledge: Further exploration of a prototype approach. *Journal of Personality and Social Psychology*, *52*(6), 1061.

Sinha, V., Lazar, A., & Sharif, B. (2016). Analyzing developer sentiment in commit logs. In *Proceedings of the 13th international conference on mining software repositories* (pp. 520–523).

Wei, T., & Simko, V. (2017). *R package "corrplot": Visualization of a correlation matrix*. Retrieved from https://github.com/taiyun/corrplot

Wickham, H. (2019). *Stringr: Simple, consistent wrappers for common string operations*. Retrieved from https://CRAN.R-project.org/package=stringr

Wickham, H. (2020). *Tidyr: Tidy messy data*. Retrieved from https://CRAN.R-project.org/package=tidyr

Wickham, H., François, R., Henry, L., & Müller, K. (2021). *Dplyr: A grammar of data manipulation*. Retrieved from https://CRAN.R-project.org/package=dplyr

Wickham, H., & Seidel, D. (2020). *Scales: Scale functions for visualization*. Retrieved from https://CRAN.R-project.org/package=scales

Xie, Y. (2015). *Dynamic documents with R and knitr* (2nd ed.). Boca Raton, Florida: Chapman; Hall/CRC. Retrieved from https://yihui.org/knitr/

Zhang, Y., Lo, D., Kochhar, P. S., Xia, X., Li, Q., & Sun, J. (2017). Detecting similar repositories on github. In *2017 ieee 24th international conference on software analysis, evolution and reengineering (saner)* (pp. 13–23). IEEE.