Mining the Importance of the Descriptive Comments in Commits

Rohith Patnamshetty
Northern Arizona University
Flagstaff, Arizona, USA
rp937@nau.edu

Sai Nikhil Edulakanti
Northern Arizona University
Flagstaff, Arizona, USA
se722@nau.edu

Venkata Susai Haneesh Vallamkonda
Northern Arizona University
Flagstaff, Arizona, USA
vv474@nau.edu

Shreyas Srinivas
Northern Arizona University
Flagstaff, Arizona, USA
ss4573@nau.edu

Abstract

In GitHub to commit any changes related to any project, the contributors are going to upload their changes and also mention their comments for the changes that they have done so that there is no miscommunication between the project owner and the contributors. If the comments were not clear and not descriptive then the project owners and other contributors may face several issues. To address and understand the importance of descriptive comments in commits we are proposing this paper. Our main aim of the paper is to show how the descriptive comment can help out the project owners and contributors. Along with this we also wanted to mention some average count of words for commit so that the comment is descriptive enough or not for the project owners and contributors. For achieving our aim, we are going to analyze five open-source GitHub repositories which have different types of contributors with their style of comments in commits.

Keywords: GitHub, open-source projects, repositories, comments, commits

ACM Reference Format:
Rohith Patnamshetty, Sai Nikhil Edulakanti, Venkata Susai Haneesh Vallamkonda, and Shreyas Srinivas. 2022. Mining the Importance of the Descriptive Comments in Commits. In . ACM, New York, NY, USA, 6 pages. https://doi.org/XXXXXXX.XXXXXXX

, ,

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

When a user contributes some changes to a project then it is called a commit. During these commits, the contributor will have a blank space to comment on their work and the changes they had made. These comments are going to be the communication between the project owners and the contributors. These comments also help us pinpoint where bugs may occur and provide insight into what the contributor is thinking. With all these advantages if the comments are left out blank or not descriptive enough then it is going to be a huge problem for any type of project. If the project is not open source, then the problem might resolve easily because the project owners know the people who are contributing to the project so they can contact the people for any problem. Now consider if the project is open source, then the project owner doesn't know the people who are contributing to the project, he also doesn't know whether to trust this person or not if the comment is left out blank or the comment is not descriptive enough. To help the project owners and other contributors what the changes you have done to the project it is very important to mention the comments in commits and the comment is descriptive enough to understand the changes developers had done.

1.1 Research Problem

If the comments in commits are not descriptive enough, then the communication between the project owners and the contributors is lost. This can be a witness in several GitHub projects which might be small or big projects, where a developer might have contributed extraordinary work to that project, but the comment is not descriptive enough to understand the work he had done and the work rejected by the project owner. Like wisely, the same will apply to the project owners too where they might reject the work which can add additional success to their projects. If we were able to resolve this issue then the communications between the contributors and project owners are not going to be last and in return, more contributors and project owners will benefit from an increased amount of accepted code commits.

1.2 Research Purpose

The purpose of this study is to understand the importance of the descriptive comments in commits by the developers. This analysis helps us to understand the minimum word count for the comment to be considered descriptive enough or not so that the developers might have an idea of how many words are required so that their changes might understand by other contributors and project owners. With this work, the communication between the project owners and contributors also increases and the acceptance rate of the commits increases by the project owners because they can understand the work which contributors had done by reading out the comments.

1.3 Research Questions

In this paper we are going to analyze the data and going to answer the following research questions:

- RQ1: What is the average word count for the commit comments?
- RQ2: What are the most commonly used keywords in commit comments?
- RQ3: How can the word count and keywords be used to determine the purpose of the commit?

1.4 Objectives

The objective of this project is to analyze the already committed comments of some of the open-source projects and whether the comments are descriptive enough or not. Along with this get out the list of some common words which are used for committing comments.

1.5 Contributions

The main contributions of this paper are to validate the conclusions of the previous related papers and how we can contribute to getting communication better between the project owners and contributors with the commit comments.

2 Related Work

The research paper on Automatically Generating Commit Messages via Summarization of Source Code Changes done by Luis Fernando Cortés-Coy, Mario Linares-Vásquez, Jairo Aponte, Denys Poshyvanyk [1] mentioned that they had analyzed 23k+ java projects and found out that only 10 percent of the commit messages were descriptive and over 66 percent of comments contains less than 15 to 20 words. This paper also evaluated the effectiveness of a software called ChangeScribe. This software helps the developers to generate descriptive comments for commits automatically with all the changes which they had done to the previous code. The main focus of this research paper is to differentiate how this software is generating comments for commits when compared with human comments. This paper mentions that there are issues with human comments such as message descriptiveness and conciseness. The statistics in this research paper suggest that descriptiveness and length of the commit messages have been problems in Java-based projects. Another research paper on The Relationship between Commit Message Detail and Defect Proneness in Java Projects on GitHub was done by Jacob G. Barnett, Charles K. Gathuru, Luke S. Soldano, and Shane McIntosh [2] investigated the relationship between the commit message details and defect proneness in java projects. Their research is mainly based on the two research questions. 1) Is there any relation between commit message volume and defect proneness? 2) Is there any relation between commit message content and defect proneness? For their first research question, they added their commit volume metric to the set of baseline metrics and performed analysis on Just in Time (JIT) models and finally, they found out that the commit message volume metric contributes up to 43 percent of the amount of the explanatory power to the Just in Time (JIT) models. According to the Just in Time (JIT) models, message volume contributes up to 25 percent to the explanation of defects and message content contributes up to 72 percent. According to their reports, it is suggested that the code issues are correlated to commit messages. So, that if we can resolve the commit message issues by encouraging descriptive comments and mentioning the specific purpose for commits might resolve the issues caused to the developers like bugs etc. A systematic literature review on Automatic code Summarization which has been done by Yuxiang Zhu, and Minxue Pan [3] describes that they analyzed 41 studies to gain an understanding of automatic code summarizing techniques. They investigated all these 41 studies and described the data extraction techniques, description generation methods, evaluation methods, and related artifacts of those works. In this paper, they tried to answer their research questions with their analysis. The research questions are 1) Which specific techniques are used for extracting the information in summarizing source code? 2) How to generate natural language descriptions and what kind of summaries can be automatically generated from the source code? 3) What evaluating procedures have been used to access the results in each paper? 4) How can we categorize source code artifacts from which we can generate natural language descriptions? Based upon their analysis of 41 studies they have found the answer to their first research question regarding specific techniques used in data extraction. The techniques used for data extraction are mainly based on information retrieval (41 percent), pattern identification (20 percent), external description mining (10 percent), natural language processing (17 percent), and machine learning (32 percent). For their second research question, they found out that description generation method distribution is classified into four methods such as template-based (46 percent), machine learning based (29 percent), term based (17 percent), and external description based (10 percent). For

their third research question, they found out that there are five evaluation method distributions as Manual Evaluation (56 percent), Statistical Analysis (39 percent), Gold Standard Summary (17 percent), Extrinsic Evaluation (12 percent), and none (10 percent). For their fourth research question, they categorized code artifacts into seven types such as method (32 percent), code segment (29 percent), code change (22 percent), class (12 percent), test case (5 percent), package (2 percent) and variable (2 percent). This paper concludes that due to the development of the software industry and the increase in software complexity program comprehension is very important so the need for automatic code summarizing techniques growing. Although this literature review topic does not match exactly to descriptiveness of commit comments it does relate to the descriptiveness of code comments and the problems associated with them transfers to commit comments.

3 Methods and Results

3.1 Methods

In the first step of our data collection, we started with the selection of the repositories where we can get the comments for the commits for our analysis. For selecting the repositories, we had followed some criteria which we mentioned below:

- Must be a public repository
- Must have at least ten contributors for that project.
- The repository must be active and have at least a commit within the past week.

or selecting the repositories, we have used GitHub search with keywords open-source and java projects. From that list, we left out the repositories which consist of other languages than English. From the English-based open-source repositories, we again classified the repositories based on the number of contributors. From that list again we classified the repositories based on their recent commits which are within a week or not. From the list of repositories that we got earlier, we started analyzing the count of the pull requests in each repository so that the repositories we will be going to select are different types such as a high number of pull requests, medium number of pull requests and a smaller number of pull requests in repositories. If all the repositories are having higher pull requests, it doesn't analyze correctly because all the repositories cannot have that high number of pull requests. It is going vice versa too if we select only medium and a smaller number of pull requests individually. For not facing these issues we have selected two smaller numbers of pull requests (less than 100 pull requests), two medium numbers of pull requests (less than 300 pull requests), and two higher numbers of pull requests (more than 300 pull requests) in the repository. From the above criteria, we picked six repositories. The selected six repositories are listed below:

- Runelite/runelite (579 pull requests)
- Rails/rails (355 pull requests)
- Facebook/react (234 pull requests)
- TheAlgorithms/Java (142 pull requests)
- Jquery/jquery (20 pull requests)
- Jenkinsci/jenkins (68 pull requests)

For the data collected from these repositories, we used GitHub APIv4 (Graph QL) for extracting the resent 50 commits where you can see the below code snippet. If the repository doesn't consist of 50 commits, then we extracted the total number of commits that are present in that repository. All these results are converted into CSV files and kept stored for future use in statistical analysis. For all these CSV files we start our

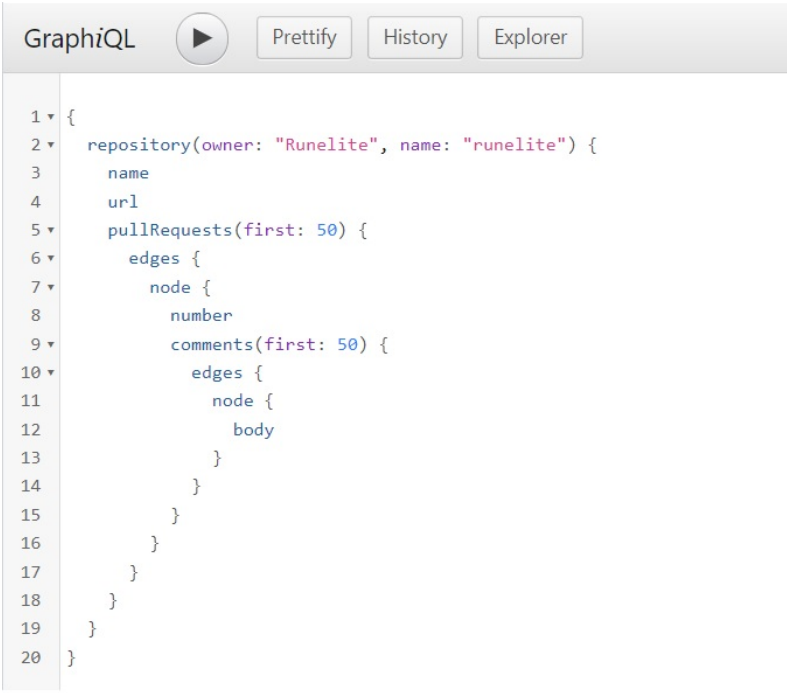

Figure 1. Screen shot of mining out a repository

statistical analysis using Python Google Colab with Pandas. In the first step, we analyze how many comments are there in CSV file, then we are going to analyze the total word count and finally average words for the comment. After getting all this, we are going to list out the top 10 common words used in the commits.

3.2 Results

In this section we are going to answer all three research questions of this paper below:

3.2.1 [RQ1] What is the average word count for the commit comments? By mining the six repositories which we had selected previously we downloaded a total of 448 comments. From that comments, we have found out that the

average word count for the commit is 26. The following table breaks down the average word count for the repository.

S.No	Repository Names	Total Comments Downloaded	Total Words in the comments	Average count of words per comment
1	Runelite/runelite	23	618	26.86
2	Rails/rails	91	1931	21.21
3	Facebook/react	127	4436	34.92
4	Jquery/jquery	111	3320	29.90
5	Jenkinsci/jenkins	74	1784	24.10
6	TheAlgorithms/Java	22	572	26.00
	Total	**448**	**12,661**	**162.99**

Table 1. The percentage of average words per comment in each repository

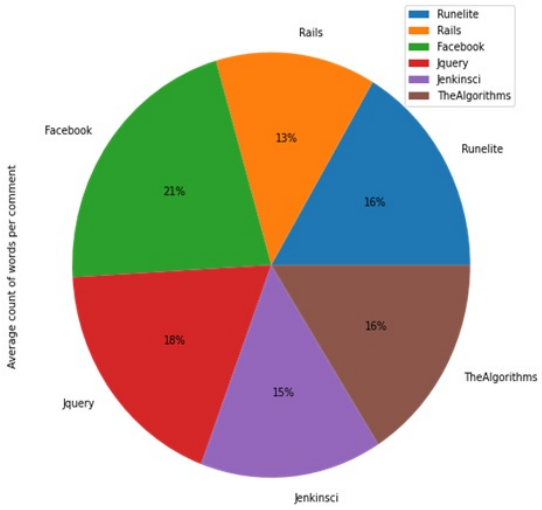

Figure 2. The percentage of average words per comment in each repository

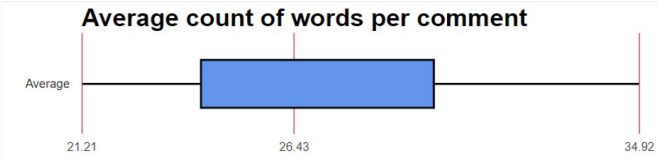

Figure 3. Average count of words per comment

In figure 2 we can understand that all the repositories have an almost similar percentage of average when compared to each of them selected. From the above steps, we can understand that the average word count in comments for a commit is 26. We can also say that if the words are 26 in the comments for a commit then the comment is descriptive enough to get the communication successful between the project owner and contributors

3.2.2 [RQ2] What are the most commonly used keywords in commit comments? From the six repositories we had selected earlier we selected one repository from that and found out the most used words for that repository. We selected the Facebook/react repository based on the maximum average count of words for comment. If the words are maximum, we can analyze and successfully analyze the third research question.

1) Code
2) I
3) You
4) We
5) Pull
6) Merge
7) Java
8) User Interface
9) Thanks
10) Work

Figure 4. Common Keywords used

Firstly, we filtered manually some of the words like 'a, 'the' and 'and'. Then we performed the analysis and found out the above words which are commonly used by eliminating prepositions and conjunctions so that we can analyze only the keywords for that repository. We have left out 'I', 'You', and 'We' because these words suggest that there was a conversation going on between the project owner and the contributors.

Some of the keywords like code, java, and User Interface suggest the application platforms which the people working on. A keyword like Thanks will be common in many of the repositories as they talk to each other and take suggestions for different things. The other keywords pull, merge and work are commonly used words in any GitHub repository as they are the GitHub keywords.

3.2.3 [RQ3] How can the word count and keywords can be used to determine the purpose of the commit? Looking at the word count and keywords from the mined repositories we can say that the commit comments are used for the conversation purposed between the project owner and all contributors. For example, when we consider the pull and merge keyword which we had found out earlier has been

one of the most common words used in the repository these words indicate that the contributor or the project owner having a conversation regarding whether his/her contribution can be merged with the project or not. We have also found out that the average word count of the committed comments is 26 which is normally greater than the word count of a normal English sentence this also indicates that the most words in the comments help out better conversations between the project owners and contributors. We also saw that from the mined repositories there was a use of emojis in the comments for the commits. We think that the emojis which has been used are a bad practice for writing a comment for the commit because the comments are the only way of communication between the project owner and the contributors.

4 Discussion

4.1 Relevance to Previous Work

Previous studies have pointed out that the commit comment lacks quality and lacks an adequate word count for being a descriptive comment. These flaws in the comments may cause errors and bugs in all the projects. In a previous paper, more than half of the 23,000+ java project comments analyzed had less than 15-20 words [1]. Our research found an average of 26 words per comment. Because we are not sure about the statistical indicators of previous studies used so it is uncertain that our result will be like the previous works. In addition, the quantity of the data collected between our study and previous studies differ significantly in the size, so it is likely to be a misleading comparison if we compare our work to the previous work.

Correlations between Java Bug and commit reports are mostly due to comment length and comment description. Even more so, assuming that comment length and meaning are positively correlated, words in comments should be more descriptive. An average English sentence is about 15-20 words, so theoretically the data set we analyzed should be more descriptive than the previous work. The comment is feasible only if the content of the comment is relevant to the project and the purpose of the commit.

4.2 Future Research

Our research for this paper was prompted by some other research questions which might require more extensive data mining skills and research ability. Our project just went only through a quantitative analysis of different comment metrics. In future research, we can include more qualitative work. We can also conduct some surveys and interviews with the developers regarding the comments in the commits so that we can also include the opinions of developers. Some of the future research questions are

- What keywords should be used by developers so that the comment is descriptive enough?
- How the commit comments are useful to developers?
- What is the thinking of the developers regarding descriptive comments in commits?

5 Threads to Validity

This section describes the main threads that might affect our results and conclusion of the research.

5.1 Scope of the data

Our paper analyzed only six GitHub repositories. This is considered a small data set when we compare it with some professionally mined research papers that analyze over 1000 GitHub repositories. Due to this, we can't generalize our conclusion to all cases. We can, however, speculate that the findings drawn from our research will most likely apply equally to all other large research done in this area.

5.2 Statistical Metrics

For the sake of our analysis, we chose mean values, which can give a summary of the data but aren't always the best viewpoint. Because we only looked at one parameter in our research, it's possible that outliers tipped the mean value in the direction of the findings. Based on the variation of the two values, the median value may have been useful in confirming or disputing the significance of the mean number.

5.3 Keyword Bias

The fact that "java" is the seventh most frequently used word is probably because our focus is just on java projects. Due to the fact that it is the corresponding extension for a Java file, this can skew the statistics to indicate this. Because they provide a route of where the problem occurred, we can presume that contributors are referring to a specific in the commit or they are inputting trace-back error messages. This information can still be informative because it sheds light on the purpose contributors use to commit comments, which is important given that we are interested in how commit comments are used.

6 Conclusion

In this paper, we studied the importance of descriptive comments in commits which can help project owners and contributors. We studied the six-open source GitHub repositories which have different types of contributors with their style of comments in commits. We aimed to answer three research questions: RQ1 focused on the average count of the commit's comments, RQ2 focused on the most commonly used keywords in commit comments, and RQ3 focused on can the word count and keywords can be used to determine the purpose of the commit. For RQ1, Our results show that by mining the six repositories we had selected previously, we downloaded a total of 448 comments. We have found out that the average word count for the commit is 26. We can also say that if the words are 26 in the comments for a commit then

the comment is descriptive enough to get the communication successful between the project owner and contributors. For RQ2, Our results show that the most commonly used keywords in the repository include Code, I, You, We, Pull, Merge, Java, Thanks, and Work. Based on these words from repositories we can say that there was a conversation between the project owner and contributors. For RQ3, Our results show that the word count and keywords from the mined repositories we can say that the commit comments are used for the conversation purposed between the project owner and all contributors. Our research can be utilized to assist developers in the future write more profound and descriptive commit comments when they contribute to a project. By writing more descriptive comments, repository owners and other contributors may find it easier to understand the purpose of the commit. For future work, we could include more repositories, analyse a larger number of commit comments, and conduct a more qualitative analysis like interviews.

References

[1] Luis Fernando Cortés-Coy, Mario Linares-Vásquez, Jairo Aponte, and Denys Poshyvanyk. 2014. On Automatically Generating Commit Messages via Summarization of Source Code Changes. *2014 IEEE 14th International Working Conference on Source Code Analysis and Manipulation* 14 (2014), 275–284. https://doi.org/10.1109/SCAM.2014.14

[2] C. K. Gathuru L. S. Soldano, J. G. Barnett and S. McIntosh. 2016. The Relationship between Commit Message Detail and Defect Proneness in Java Projects on GitHub. *MSR '16: Proceedings of the 13th International Conference on Mining Software Repositories* (2016), 496–499. https://doi.org/10.1109/MSR.2016.063

[3] Yuxiang Zhu and Minxue Pan. 2019. Automatic Code Summarization: A Systematic Literature Review. *arXiv* (2019). https://doi.org/10.48550/ARXIV.1909.04352