Studying the Bias of Evaluations of Code Review Recommendation Systems

Ian Gauthier
Faculty of Software Engineering, McGill University, Canada
ian.gauthier@mail.mcgill.ca

Abstract- Code review has become an integral part of the software design lifecycle. Many tools have been implemented and tested to attempt to automate the process of choosing the correct reviewer for a given change. However, there has not been a study performed to assess how tests of these software systems are biased. We plan to perform a study on the Gerrit code review database to perform this assessment and answer more definitively how successful automated recommendation systems are in practice.

I. Introduction

Code review of software systems represents a stop gap in software development used to ensure that any change added to the final system will improve the quality of a system as a whole. It has been found that consistent use of code review leads to less post-release bugs in comparison to systems with less coverage[4]. As such it has been implemented by many software teams both in private use and in open source systems. This can lead to the question of how to choose the person who is best suited to reviewing a specific change made to a code base. Bacchelli et al. [1] found that code review is more successful overall when the reviewer of the change has prior knowledge of the code that they are reviewing. As such, it would behoove teams to attempt to ensure that the reviewers that they assign to a given change are prepared to assess the change, giving them a better chance to successfully and efficiently review the work. There have been many attempts to automate the process of choosing the best reviewer for the job, often using historical data about the users past reviews to choose the correct person [3][5][6]. Often, these recommendation systems have been tested to understand how well the system performed at choosing the correct reviewer for the job[3][5][6]. The baseline that is used is often a comparison between the reviewer chosen by the recommender system and the reviewer that was actually assigned to check the change. However, there has not been a study performed to check how these evaluations have been biased. Specifically, the question we pose is how often were the recommendations that were deemed correct due to the reviewer chosen by the recommender system being the reviewer who actually checked the code not actually correct as the reviewer did not feel they were the right person for the job. In addition, we pose the question of how often did recommendations deemed incorrect by the same methodology actually provide a reviewer that would have been prepared to review the given change. In this study, we will attempt to answer these questions and thus provide a better understanding of how biased evaluations of recommender systems are in practice.

Paper organization: The rest of the paper will be organized as follows: Section II will provide an overview of the study. Section III will state the research questions and the methodology being used to answer them. A timeline of the project is presented in Section IV. Section V provides conclusions.

II. Study Design

This section will overview the methodology of the study which will be implemented to assess how evaluations of recommender systems that rely on historical data are biased. First, we will discuss the corpus of data which will be collected and used to implement the recommendation system. Second we will overview the replication of the recommender system created by Zanjani et al. Lastly, we will overview the study we will perform to assess the bias of recommender system evaluations.

A. Corpus of Review Data

In order to obtain information on which reviewers were recommended to specific change to review, we will first need to mine a dataset to perform this process. In this project, we will be using change and review data provided by Gerrit[2] for the Gerrit codebase itself. We will poll the change and review data provided by the Gerrit API to find all of the changes that have been reviewed by another user. Along with the list of changes, we will extract the review comments for each change and the list of files each reviewer worked on to perform each specific review as well as the dates on which reviews were performed. This data will be used to perform our analysis on which reviewer is the most prepared to undertake the review of each change based on the historical data of the system.

B. Creation of Recommender System

In order to test how well recommender systems that use historical records to obtain a suitable reviewer for a change, we will need to create a system that mimics the state of the art in this field. There are several studies that have attempted to tackle this task [3][5] but we will replicate the model created by Zanjani *et al.* [6] which is currently the best performing model available. Mimicking the best model which is currently available will allow us to state with confidence that we are achieving the highest level of success possible from a historical based system.

C. Study of Success Rate of Recommender System

Once the model has been created, we will run the system on the Gerrit data in order to find what reviewers the system would have recommended to review the changes made to the Gerrit system. Once this has been completed, we will contact the reviewers who were recommended in order to see whether they believe they were the correct programmer for the job. We will contact all assignees who were chosen by the recommender system and were also the ones who performed the review in reality and ask them if they feel they were the correct user to assign. We will also contact reviewers who our model assigned to the change but did not end up performing that specific review and ask them if they feel they would have been able to perform it well. From the responses we receive to these questions we will be able to assess how correct the normal metric of comparing the person was suggested for a change by the system and the person who actually performed the review actually is.

III. Research Questions

We define two research questions below in order to clearly outline the goals of the project being undertaken.

RQ1) Pessimism: How often were past assignees suboptimal?

This question attempts to quantify how frequently a suggestion is deemed correct due to the reviewer who is chosen by the system being the same as the reviewer who actually reviewed the change is not actually a good suggestion.We will contact each reviewer for which our tool chose the reviewer who actually ended up reviewing the change and ask them if they felt they were the correct user to assign. We will then report what percentage of these reviewers felt that they were the optimal reviewer and the percentage who felt they were not well prepared to review said change.

RQ2) Optimism: How often are incorrect recommendations actually appropriate?

In the case that our review system does not choose the reviewer who ultimately performed the review, we will contact the user who our model decided was the best person for that change. We will interview them to attempt to answer whether they felt that they would have been able to perform the review had they been given the opportunity. We will then report results on what percentage these users were actually a correct choice for the model but were not afforded the opportunity to review the change.

IV. Project Timeline

In this section, we will define the timeline for the project. This will be divided into two sections: (1) An initial study that is smaller in scope which is intended to complete the requirements of the course; and (2) A more expansive study on a larger set of data which will produce a full-scale research paper. This will include a larger number of projects than the Gerrit dataset which will be the basis of the original study.

A. Initial Study

Here, we present the milestones of the study using only the Gerrit project data:

- Data Extraction (12 February 2020): Extraction of the necessary data from the Gerrit corpus.
- Creation of Recommendation System (29 February 2020): Complete software program and begin to contact reviewers.
- Project Presentation (7 April 2020): Present findings of the initial study.
- Project Report (17 April 2020): Report full findings of the initial study.

B. Full Study

In this section we will discuss the expansions to be made to the original study to improve upon its findings.

- Data Extraction (15 May 2020): Extraction of data from larger corpus of projects.
- Begin Communications (30 May 2020:) Start process of contacting reviewers for feedback.
- Report (31 July 2020): Submit full-scale findings based on multiple projects.

V. Conclusions

Many studies have been performed to assess the success of recommender systems in proposing the correct reviewer for a specific change. However, there has not been a study performed to assess how these evaluations have been biased. In this proposal, we describe our plan to study how these historical based evaluations of code review recommendation systems are biased.

References

[1] A. Bacchelli and C. Bird, “Expectations, outcomes, and challenges of modern code review,” *2013 35th International Conference on Software Engineering (ICSE)*, 2013.

[2] “Gerrit Code Review,” https://www.gerritcodereview.com/

[3] H. Kagdi, M. Hammad, and J. I. Maletic, “Who can help me with this source code change?,” *2008 IEEE International Conference on Software Maintenance*, 2008.

[4] S. Mcintosh, Y. Kamei, B. Adams, and A. E. Hassan, “The impact of code review coverage and code review participation on software quality: a case study of the qt, VTK, and ITK projects,” *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*, 2014.

[5] P. Thongtanunam, C. Tantithamthavorn, R. G. Kula, N. Yoshida, H. Iida, and K.-I. Matsumoto, “Who should review my code? A file location-based code-reviewer recommendation approach for Modern Code Review,” *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, 2015.

[6] M. B. Zanjani, H. Kagdi, and C. Bird, “Automatically Recommending Peer Reviewers in Modern Code Review,” *IEEE Transactions on Software Engineering*, vol. 42, no. 6, pp. 530–543, Jan. 2016.