2019 5th International Conference for Convergence in Technology (I2CT)
Pune, India. Mar 29-31, 2019

Review on Textual Data Mining for Reviewer Recommendation in Pull-Based Distributed Software Development

Raveen Savinda Rathnayake
College of Computer Science and Engineering
University of Westminster
London, UK
w1610086@my.westminster.ac.uk

Guhanathan Poravi
Department of Software Engineering
Informatics Institute of Technology
Colombo, Sri Lanka
guhanathan.p@iit.ac.lk

Abstract— **Distributed Software development process has dramatically changed over the last decade due to the integration of social collaborative development environment. The pull-based software development methodology made its mark in the open source distributed development as it is a convenient and effective system to organise collaborative contribution. Code reviews for software projects have been a best practice in software engineering. With the emerge of pull-based software development methodology, code reviewers faced difficulty in reviewing the contributions because of the higher number of incoming pull requests. In order to address this problem, reviewer recommendation systems have been implemented. In these systems, textual data mining techniques have been applied. This paper focuses on identifying the different approaches in terms of textual data mining used in the domain of the reviewer recommendations in pull-based software development and identifies their drawbacks and room for improvement. This paper contains the initial part of ongoing research and in the future, we hope to use this knowledge to come up with a solution that addresses the identified drawbacks and the identified improvements.**

Keywords—Pull Request, Text Mining, Machine Learning, Reviewer Recommendation, Distributed Software Development

I. INTRODUCTION

In this section, we present a brief introduction to the pull-based development methodology. In section 2, we provide the background with an analysis of the contribution process of the pull-based development mechanism, followed by the problem which states what we focus on this paper. Section 3 presents the setup we used to capture the various aspects in terms of knowledge for the analysis of this paper. Section 4 provides the related work and existing solutions. Section 5 evaluates the existing solutions and provides what has been missing in the related work and finally, in section 6, the paper is concluded with future work.

The distributed Software development process has dramatically changed over the last decade due to the integration of social collaborative development environment [1]. The pull-based software development methodology made its mark in the open source distributed development as it is a convenient and effective system to organise collaborative contribution [2]. Pull-based development provides a great advantage in terms of the process automation compared to the traditional open source collaborative mechanisms such as patching through mailing lists or Bugzilla [3]. Nowadays social coding community platforms such as GitHub, Bitbucket, GitLab are using this pull-based model in their platforms.

II. BACKGROUND

Due to the lower entry barrier for the contributions, the pull-based methodology integrated social code hosting sites notably GitHub, became popular among developers. The contribution process in a pull-based software development environment can be done via a pull-request (PR). An external contributor can contribute to a project hosted in one of the above-mentioned social coding platforms by sending a PR which contains all the code changes. The summarized version of the contribution process takes place in a pull-based social code platform is shown in Figure 1. In general, Core team member, Contributor and Commenter are the three main roles involves in a contribution process. A core team member is the one who decides whether to accept or reject a PR and has the permission to directly commit code changes to the project repository. The contributor is the one who does the contributions and Commenter can add comments to the PR often suggesting improvements. A core team member can also leave comments on the PR. Hence, commenters and core team members can also be named as reviewers as well.

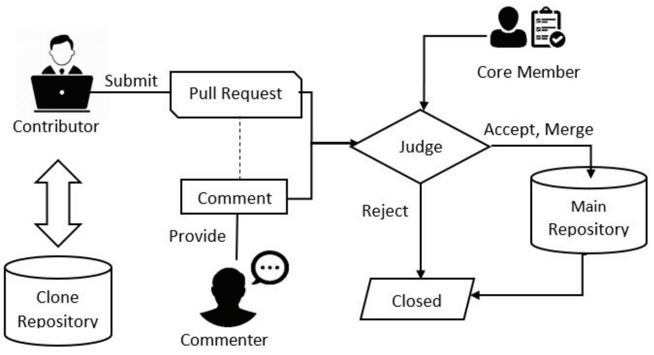

Figure 1: Summarised contribution process

The contribution process starts with a contributor forking the main project repository. Then contributor adds contributions to the forked local repository and sends a PR to the main repository which contains all the local code changes. After that core members and commenters can go through the PR and add suggestions and improvements as comments. Based on the comments received, then the contributor makes changes and pushes them to the existing PR. Finally, a core team member will decide whether to accept these changes and merge the PR to the main repository or reject and close the PR. Figure 2 illustrates an actual PR evaluation. This PR with id 34511 was sent to GitHub hosted *Ruby on Rails* project repository. The contributor *TomSpencerLondon* made the PR and core team member *gmcgibbon* approved and merged the PR.