The Journal of Systems and Software 102 (2015) 109–122

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

A time-based approach to automatic bug report assignment

Ramin Shokripour[a,*], John Anvik[b], Zarinah M. Kasirun[a], Sima Zamani[a]

[a] *Faculty of Computer Science & Information Technology, University of Malaya, Kuala Lumpur, Malaysia*
[b] *Department of Computer Science, Central Washington University, Ellensburg, WA 98926, USA*

ARTICLE INFO

Article history:
Received 17 April 2014
Revised 25 December 2014
Accepted 26 December 2014
Available online 31 December 2014

Keywords:
Term weighting technique
time metadata
Tf-idf Technique

ABSTRACT

Bug assignment is one of the important activities in bug triaging that aims to assign bugs to the appropriate developers for fixing. Many recommended automatic bug assignment approaches are based on text analysis methods such as machine learning and information retrieval methods. Most of these approaches use term-weighting techniques, such as term frequency-inverse document frequency (tf-idf), to determine the value of terms. However, the existing term-weighting techniques only deal with frequency of terms without considering the metadata associated with the terms that exist in software repositories. This paper aims to improve automatic bug assignment by using time-metadata in tf-idf (Time-tf-idf). In the Time-tf-idf technique, the recency of using the term by the developer is considered in determining the values of the developer expertise. An evaluation of the recommended automatic bug assignment approach that uses Time-tf-idf, called ABA-Time-tf-idf, was conducted on three open-source projects. The evaluation shows accuracy and mean reciprocal rank (MRR) improvements of up to 11.8% and 8.94%, respectively, in comparison to the use of tf-idf. Moreover, the ABA-Time-tf-idf approach outperforms the accuracy and MRR of commonly used approaches in automatic bug assignment by up to 45.52% and 55.54%, respectively. Consequently, consideration of time-metadata in term weighting reasonably leads to improvements in automatic bug assignment.

1. Introduction

Bug triaging, an important process in software maintenance, has a significant effect on the quality of software projects (Jeong et al., 2009). In this process, a project member called the *triager*, who may also be one of the developers, investigates the validity of the reported bugs. Valid bugs are then assigned to the most appropriate developer(s) for fixing. The process of assigning the bugs to developers is called bug assignment. Traditional manual bug triaging is time-consuming and tedious. More importantly, bug triage imposes extra costs on projects (Anvik and Murphy, 2011). Accordingly, various research has sought to improve this process by using (semi)automatic bug assignment processes (e.g. Anvik and Murphy, 2011; Kagdi et al., 2012; Servant and Jones, 2012; Tamrawi et al., 2011).

Machine learning (ML) methods (Ahsan et al., 2009; Anvik and Murphy, 2011; Bhattacharya et al., 2012; Cubranic and Murphy, 2004; Nasim et al., 2011; Tamrawi et al., 2011) and information retrieval (IR) methods (Kagdi et al., 2012; Linares-Vasquez et al., 2012; Lucca, 2002; Matter et al., 2009; Moin and Neumann, 2012; Nagwani and Verma, 2012; Zhang and Lee, 2013) have been widely used by researchers in bug assignment approaches. These approaches use textual information which was extracted from software repositories, to assign developers to new bugs. Although recent non-text based approaches are emerging to address the limitations of text analytic methods (Panichella et al., 2013), text-based methods are still the most-used and effective techniques in this area (Sun et al., 2014).

Term similarities between a new bug report and information resources are used for establishing a relationship between the new bug report and activities of the developers. The importance of each term in establishing this relationship is determined using a term-weighting technique. The most commonly used term-weighting technique for automatic bug assignment is *term frequency-inverse document frequency* (tf-idf) (Cavalcanti et al., 2014). A significant difference between the textual resources of software projects and documents in other areas (e.g. newspapers) is the existence of metadata, specifically the time of using the term.

This paper presents ABA-Time-tf-idf, an automatic bug assignment approach using the Time-tf-idf term weighting technique. ABA-Time-tf-idf improves the accuracy of a bug assignment approach. Compared to other approaches, this new approach is lightweight, as it only includes a time-based term weighting technique and a simple ranking method to rank developers based on their expertise. To the best of our knowledge, this approach is unique.

An evaluation against other commonly used bug assignment approaches across three different datasets shows that the ABA-Time-tf-idf approach obtains better performance than the comparable ML,

* Corresponding author. Tel.: +60 17372 4849.
E-mail address: rshokripour@gmail.com (R. Shokripour).

http://dx.doi.org/10.1016/j.jss.2014.12.049
0164-1212/© 2014 Elsevier Inc. All rights reserved.

IR, and ABA-tf-idf approaches. ABA-Time-tf-idf outperforms the SVM, Naive Bayes, VSM and SUM approaches by as much as 45%, 28%, 11% and 19%, respectively. The new approach also improves on the mean reciprocal rank (MRR) by up to 55.54%, 30.88%, 15.37% and 25.03%. Finally, ABA-Time-tf-idf also outperforms the accuracy and MRR of ABA-tf-idf by up to 11.8% and 8.94%.

The rest of this paper proceeds as follows. First, the motivation of this research is presented in Section 2. Next, the details of the proposed method are described in Section 3. The empirical evaluation of the method is presented in Section 4 and the evaluation results and analysis are presented and discussed in Section 5. Section 6 discusses the threats to validity. Then, in Section 7, the most related works in bug assignment are investigated. Finally, the paper is concluded in Section 8.

2. Motivation

Bug assignment approaches which use text analysis techniques use the term similarity between the new bug report and software information resources, such as bug repository and source code, to establish a relationship between the new bug report and developer expertise. Recently, researchers have augmented the text analysis portion of such approaches with metadata. Examples of the use of metadata include the product and component of the bug (Xia et al., 2013), bug tossing behavior (Bhattacharya et al., 2012; Jeong et al., 2009), developer communication social networks (Wu et al., 2011; Zhang and Lee, 2013), and source code locations of the bug (Kagdi et al., 2012; Linares-Vasquez et al., 2012; Servant and Jones, 2012). This means that the accuracy of these approaches depends on the accuracy of the text analysis technique. Therefore, improving the accuracy of the text analysis technique could result in improving the accuracy of the bug assignment approach. This paper presents a new bug assignment approach that uses the time when the terms were used by the developer in the term weighting step of the text analysis process.

Other approaches have also used time-metadata, such as the time spent fixing previous bugs (Nguyen et al., 2012) and the time distance between the last activity of developer and the reporting date of the new bug (Bhattacharya et al., 2012; Kagdi et al., 2012), to improve the results of the text analysis methods. However, time-metadata have been used by the other techniques, we used time-metadata to determine the weight of the terms used by the developers. The proposed term weighting technique in this paper is referred to as time-based tf-idf (Time-tf-idf). An advantage of this technique is that it removes the need for activity thresholding (Anvik and Murphy, 2011; Matter et al., 2009), whereby developers who have been inactive for more than the threshold are removed from the developer list. Such pruning of the data may result in the loss of useful information. Time-tf-idf addresses this issue by giving lower weights to the earlier activities, but not removing them. In other words, early activities which will have less effect in determining the expertise are given lower weights than recent activities, as recent activities will have a greater effect in determining developer expertise. This is due to the use of the time difference between the terms used in each activity and those used in a new bug report. The larger the time difference, the smaller the term weight for the terms in that activity.

Moreover, projects have different goals or address different requirements at different points in time (Gómez et al., 2009), and the term weights should reflect this situation. For example, a developer that worked on a feature of the project five months ago has more potential to resolve a new bug related to this feature than the developer who worked on the same feature two years ago. Therefore, the last time that a developer used a keyword associated with that feature is a suitable parameter for assessing the developer's current expertise relative to the feature. Furthermore, developers change their expertise, such as moving from one component or product to another component or product. Accordingly, the vocabularies that the developers use also change to reflect changes in their expertise. Considering the time at which a developer uses terms may therefore improve developer recommendation accuracy.

3. Proposed approach

The proposed approach in this research seeks to improve automatic bug assignment by using the terms' time-metadata as an effective parameter in weighting the terms in the tf-idf term-weighting technique (time-based tf-idf). The tf-idf technique only deals with the frequency of the terms in the document and corpus to determine their value. However, time-based tf-idf (Time-tf-idf) also considers the time of using the term in the project in determining the weight. Fig. 1 shows an overview of the recommended automatic bug assignment approach (ABA-Time-tf-idf approach), that uses the Time-tf-idf term weighting technique. The ABA-Time-tf-idf approach has three stages: corpus creation, expertise determination, and developer recommendation.The details of these steps are described in the following sections.

3.1. Corpus creation

Regarding the usage of the time-metadata in the proposed approach, the corpus creation step is different from the corresponding step in the commonly used text analysis methods. In ABA-Time-tf-idf, the required data is collected from the version control system (VCS), a software repository for managing changes to source code and other relevant project documents. From the source code, the identifiers are extracted and used to establish the relationship between a new bug report and developer activities in the project. Therefore, each identifier in the project's source code is associated with a developer and each identifier-developer relationship represents a developer activity.

Source code identifiers are used to name the entities of the source code (such as files, classes, and methods). In this study, the names of classes, methods, fields, and parameters of the methods are extracted, as they have been found to contain more useful and less noisy data. Identifiers are often the concatenation of a set of terms used to indicate the functionality of the source code entity (Abebe and Tonella, 2010). Therefore, the extracted identifiers are also decomposed into their components using the identifier tokenization method recommended by Butler et al. (2011). Such an extracted identifier is hereafter referred to as a "decomposed identifier". The method of Butler et al. first decomposes the identifiers using common rules of term decomposition. Examples of such rules are decomposing based on the internal capitalization and decomposing based on separator characters. In addition, their approach improves tokenization by increasing the accuracy of tokenizing single identifiers and cases where identifiers contain digits. This is done by using a set of heuristics and other methods, such as word recognition and a recursive algorithm for finding the project's vocabulary in the identifiers. In this study, both the complete and decomposed identifiers are used in representing developer activity.

After extracting the identifiers, they need to be associated with the corresponding metadata. For each extracted identifier, the name of the developer who is associated with the identifier and the time stamp of the commit into the source code repository are extracted for use in term weighting. The commit time is used as the creation time of the identifier. Consequently, it is possible that a term which is used in more than one identifier of a file has various times associated with it in the generated dataset.

The results of Capobianco et al. (2013) and our recent investigation (Zamani et al., 2014) demonstrate that only using the noun terms of the decomposed identifiers significantly reduces the volume of the dataset and also the noisy data without decreasing the