Journal of Electrical and Computer Engineering Innovations
(JECEI)
Journal homepage: http://www.jecei.sru.ac.ir

Research paper

Using Machine Learning Methods for Automatic Bug Assignment to Developers

M. Yousefi[1], R. Akbari[2,*], S. M. R. Moosavi[3]

[1] *E-Learning College, Shiraz University, Shiraz, Iran.*

[2]*Department of Computer Engineering and Information Technology, Shiraz University of Technology, Shiraz, Iran.*

[3]*Department of Computer Science, Engineering, and IT, Shiraz University, Shiraz, Iran.*

Article Info

Article History:
Received 07 September 2019
Reviewed 04 November 2019
Revised 12 December 2019
Accepted 12 March 2020

Keywords:
Automatic bug assignment
Bug reports
Bug clustering
Similarity criteria

[*]Corresponding Author's Email Address:
akbari@sutech.ac.ir

Abstract

Background and Objectives: It is generally accepted that the highest cost in software development is associated with the software maintenance phase. In corrective maintenance, the main task is correcting the bugs found by the users. These bugs are submitted by the users to a Bug Tracking System (BTS). The bugs are evaluated by the bug triager and assigned to the developers to correct them. To find a related developer to correct the bug, recent developers' activities and previous bug fixes must be examined. This paper presents an automated method to assign bugs to developers by identifying similarity between new bugs and previously reported bug reports.

Methods: For automatic bug assignment, four clustering techniques (i.e. Expectation-Maximization (EM), Farthest First, Hierarchical Clustering, and Simple Kmeans) are used where a tag is created for each cluster that indicates an associated developer for bug correction. To evaluate the quality of the proposed methods, the clusters generated by the methods are compared with the labels suggested by an expert triager.

Results: To evaluate the performance of the proposed method, we use real-world data of a large scale web-based system which is stored in the BTS of a software company. To select the appropriate algorithm for the clustering, the outputs of each clustering algorithm are compared to the labels suggested by the expert triager. The algorithm with closer output to the expert opinion is selected as the best algorithm. The results showed that EM and FarthestFirst clustering algorithms with 3% similarity error have the most similarity with the expert opinion.

Conclusion: the results obtained by the algorithms show that we can successfully apply them for bug assignment in real-world software development environments.

Introduction

Software systems enter into the maintenance phase after delivery to the customer and evolve over time. Software is constantly changing due to new changes needed by the customer and fixing possible bugs. Much of the cost of software development is spent on maintenance. Since software bugs are inevitable, it is imperative to assign the bug to a proper developer.

When a bug is reported in the software, the bug must be triaged. Bug triage is an important process in the software maintenance phase and has a major impact on software quality [1]. In the triage process, the person

known as the triager, examines the accuracy of the reported bug. Valid bugs are then assigned to a developer to be fixed. The traditional and manual triage process is time-consuming and costly and imposing more cost on the project [2]. In large-scale software projects, due to a large number of developers and the possibility that they may work parallel in various project modules, finding the appropriate developer is a difficult task and it is time-consuming and inaccurate to make the necessary checks [1], [2]. For example, the large number of bug reports or the wrong assignment of a bug slows down the debugging process. In this case, automatic bug assignment and clustering of bugs based on their similarities can make the triage and bug assignment more accurate and faster. Bugs in large software systems are maintained in BTS [3].

In large software projects, 50 to 60 bug reports are saved daily in the BTS [4]. As an example, for the Eclipse project, an average of 37 bug reports is logged daily in the BTS, which requires 3 person-hours per day for manual bug triage [5]. According to the study reported by Jeong et al., 44% of bugs have been assigned to the wrong developer after the first assignment [1]. To cope with this problem, in recent years, different types of methods have been proposed by authors [2]-[6]. These researchers were aimed to automate the bug triaging process. Some of the bug triage approaches are based on text categorization [2]. However, these methods suffer from poor quality reporting and cause to assign bugs to wrong developers [6], [7]. The main task in the bug assignment is to find the appropriate developer to fix the bug by analyzing the bug history that occurred in the software. In this paper, an automated method for assigning the reported bug to the developer is presented in a closed source web-based software system. The method use clustering techniques to cluster the bugs. An expert opinion is used for accurate verification of the clustering algorithm and the outputs of each algorithm that are closest to the expert opinion are selected as the appropriate clustering algorithm. The main contributions of this paper are:

- Aggregating required data for bug triaging and assignment in a Closed Source Project (CSP).
- Using the proposed method for bug triaging in a real-world large scale web-based system.
- Adapting different machine-learning methods for data clustering and studying their performance for real-world data.

The remaining of this paper is organized as follows: the next section presents the previous works on bug triaging and bug assignment. The details of the proposed method is presented in Section “Methodology”. Section “Evaluation and Results” contains performance analysis and experimental results. Finally, conclusions and future works are given.

Related Work

In the maintenance phase, for bug triaging and bug assignment, many researchers use different information retrieval and machine learning methods to analyze textual sources in software repositories. More precisely, information retrieval and machine learning techniques have been extensively used by researchers to improve assigning bugs to developers. In this section, we review some of these methods for automatic bug assignment and bug triaging.

In [8], some bug assignment methods have been proposed. Different data sets and different input parameters have been used to evaluate the proposed method. According to this article, the number of different methods available for triage and bug correction confuses researchers. Therefore, in this paper, the work done to fix the bug is managed in a structured way. For this purpose, a structured combination of bug-solving methods is provided. Also, various aspects of bug correction are described and 6 related research questions in 5 dimensions are examined. To create infrastructure and organize bug assignment methods, 60 articles have been reviewed and classified. This study helps researchers to choose the right tools to fix the bug.

Limsettho et al. [9] presented a method for categorizing bug reports using topic modeling and two clustering algorithms. The proposed method has three phases. In the first phase, the bug reports are preprocessed and converted to topic vectors. These vectors are clustered in the second phase. Finally, each category of bugs is labeled. Alenezi et al. proposed a method to reduce the bug triage time and automatic bug assignment to a related developer. They used Naive Bayes (NB) classifier to build a predictive model that can be used to assign a new bug report to a developer in the future. Five selection methods (LOR, X2, TFRF, MI, and DFS) have been used to reduce the size of the dimensions of terms and improve accuracy. This approach has two main steps. 1) A classification model is created using reduced terms to predict an experienced developer to fix newly reported problems. 2) Redistribute the load of overloaded developers. The evaluation was performed using four reported bugs from actual projects. Precision, recall, and F-score criteria were used to evaluate the performance of the classification [10]. The implementation of a recommendation system that was parallel and scalable and based on deep learning has been presented by Florea et al. [11]. Two deep learning categories have been used: Convolutional and Recurrent Neural Networks (CNN and RNN). The main theme of this article is not about running time, but about the scalability of the system on a cluster. This is measured using the speed

criterion (the ratio of the sequential execution time to the parallel execution time) and the parallel evaluation (speedup divided by the number of processors/cores) [11]. Shokripour proposed a method that uses textual information of the reported bugs in the bug repository to assign a new bug report to a developer. This method uses the term frequency-inverse document frequency (TF-IDF) term weighting technique. By using time metadata as an effective parameter in term weighting in term frequency-inverse document frequency, an attempt has been made to improve the automatic attribution of bug. The last time the term is used by the developer is used in the assignment [12].

In another work, Shokripour et al. presented a method based on Information Extraction (IE) techniques for bug assignment in large scale open source projects (OSP) [13]. The proposed method applied on three projects and more than 41% accuracies obtained.

Guo et al. presented a method based on convolution neural network (CNN) and developer activities for bug triaging [14]. They used CNN along with batch normalization and pooling to learn from the vectors generated by Word2vec. The performance of their method was evaluated on three open-source projects (OSPs). The bug assignment problem has been considered in [15] by employing programming keywords in the bug description as well as the recent expertise of developers. The authors applied their method on 93k bug-report assignments from 13 popular GitHub projects.

Zhang and Lee have proposed a method based on the combination of an experienced model and a probability model. First, a fixed bug that similar to new bug reports are extracted using the Smooth Unigram Model (SUM). Then an experienced model and a probability model based on similar bug reports are created. To create a probability model, social networking techniques are used to determine the relationship between developers from comments in the bug reports. The experience model is then created based on a series of project activity factors in the project such as the number of bugs which is fixed by the developer. Eventually, two models are combined and a developer rating is extracted that is used for new bug reports [16]. In other work, a machine learning-based approach was proposed that uses the nearest-neighbor algorithm to classify bug reports. The method consists of two components. The first component uses the VSM method with TF-IDF weighting to convert the fixed bug report text to the term vector space and determine the similarity of bug reports to new bug reports. The second component uses social network metrics to rank developers so that a ranking list is created based on the records of developer participation in discussing similar bug reports [17].

Kashiwa used mathematical programming for bug assignment [18]. He presented an optimization method called Release Aware and Prioritized Bug Fixing Task Assignment Optimization (RAPTOR). The purpose of this method is to mitigate the task concentration and increasing the number of bugs that developers can fix.

The application of ensemble methods has been studied by Goyal and Sardana in [19]. They used five ensemble methods called Bagging, Boosting, Majority Voting, Average Voting, and Stacking. For designing these ensembles, 25 different machine learning classifiers have been used by the authors. They applied these ensembles on three OSPs. Their results showed that the ensemble methods provide better performances in comparison with the base classifiers. In [20], an algorithm based on the Developer's Expertise Score (DES) for Bug Tossing Length (BTL) has been provided. The strategy is done in two steps: The first step is an offline process for obtaining a DES, which is calculated based on priority, adaptability, and average fixed time in developer activities. The online system process involves finding capable developers using three similarity calculation criteria (feature-based, cosine similarity, and Jacquard). The second step in the online process is to create points. Hit-ratio and reassignment accuracy are used to evaluate performance. In this method, the system is compared with ML-based debugging methods using three types of classification algorithms: Navies Bayes, Support Vector Machine (SVM), and C4.5 paradigms. By testing 41622 bug reports related to Mozilla, Eclipse, Netbeans, Firefox, and Freedesktop projects, the proposed method has an average accuracy of 89.49%, the precision is 89.53%, the recall rate is 89.42% and the F-score is 89.49%, which reduces BTL to 88.5%, which shows 20% improvement over existing technologies [20].

In [21], the main goal is to create a classifier to classify the reported bugs into two predefined classes: corrective report (defect fixing) and perfective report (major maintenance). This allows the maintainers to understand the bug more quickly when new bugs are reported and to provide the resources needed to fix the bug. For this purpose, the proposed method is based on a set of specific features that are based on the occurrence of specific keywords. This set is fed to some classification algorithms to create a classification model. The results of the proposed method are based on 3 different open source projects with an average accuracy of 93.1% with classification using the SVM classification algorithm [21].

The bug assignment problem in a CSP has been considered in [22], the goal was to reduce the bug assignment time to a developer with a related specialty that is reduced by tossing length. The development of such a technique is especially challenging for closed source projects. In this paper, a score is created to identify and rank an expert developer independent of

the nature of the project. Two criteria are presented based on developer expertise and bug importance score. These two criterion are calculated using information obtained from the components and content of the bug report. To validate the proposed method, the bugs that have been reported in a CSP developed by XYZ, pvt. Ltd has been used. The result obtained for the proposed method on the selected data set has been predicted with an accuracy of 88.9% .

In [23], a method for simultaneous bug triage was proposed for the developer and the development team using two-output neural network structure (called Dual DNN). This simultaneous is used using assignments made to the developer by team classes. A multi-label classification method has been used for two outputs for learning. A combination of exploratory labels that become a function of probability has been used. First, a two-step learning plan is used, in the first step of learning a part of the team is trained, and then the communication training between the team-developer and the developer-bug is done. The scheme is designed to encode team and developer relationships based on an organizational chart, which reinforces this model of organizational change because it can be adapted to role changes in an organization. A method called KSAP (K-nearest-neighbor search and heterogeneous proximity) was proposed by Zhang et al. to automatically assign a bug to the developer using historical bug reports and a heterogeneous network of bug repository [24]. When a new bug is reported, the bug is assigned to the developer in two phases. The first phase is to find similar bug reports to the new bug using the K-nearest-neighbor (KNN) method, and the second phase is to find developers who have participated in similar bugs using Heterogeneous proximity. An experiment on the Mozilla, Eclipse, Apache Ant, and ApacheTomcat 6 projects concluded that the KSAP method could improve the bug assignment recall between 7.5% and 32.25% compared to similar new methods [25].

Lee et al. reported that most previous studies focused only on OSPs and did not consider deep-learning techniques [25]. The Convolutional Neural Network (CNN) from the machine learning branch and Word2Vec from the word embedding branch has been used for automatic bug triage. The results obtained from the proposed method on the industrial project and open-source show the advantages of the approach. In fact, by using deep learning, the automatic assignment of a bug is performed on an industrial project. The performance advantages of the proposed method have been measured in comparison with human triage in terms of accuracy and simultaneous overhead. According to bug reports for industrial projects, we simulate the situations in which the proposed system is used and confirmed the effectiveness of the proposed system [25].

A two-phase method that used the Association Rule Mining (ARM) and X-Menas algorithm was proposed by Sharma and Singh for bug triaging [26]. In the first phase, the Apriori algorithm was used to predict the assignment of new bugs. The second phase used X-Means clustering along with ARM in each cluster. The performance of the proposed method was studied on some open source projects.

Mahendran proposed an approach that uses chart databases to calculate points for engineers and assign bugs to them [27]. This method is preferred over machine learning methods because there is no need to process of extracting, analyzing, or synchronizing data. The whole database for bug management can be in the graph database, and the method can be implemented directly on bug management tools. The proposed method controls the automatic assignment of errors along with workload balancing for engineers. Graph databases manage data internally as graphs and make relationships available as ready-made graphs in the database. It is possible to identify suitable maintenance engineers with queries without any specialized tools or extraction process. Lee et al. proposed a two-phase method for cost-aware clustering of bug reports by employing the Genetic Algorithm (GA) [28] as an optimization algorithm. In the first phase of their method, a set of groups is created based on the similarities between bugs. The second phase constructs the clusters by grouping similar reports. The method was examined on the bug reports of Mozila's Firefox project.

A short survey of the previous methods is presented in Table 1. The second column shows the method used by the authors mentioned in column one. The third and fourth columns show the name and type of dataset used to evaluate the proposed methods.

As can be seen from Table 1, in most of the studies, data of OSPs have been used by researchers and there are a few works that have considered CSP data sets. The main OSPs that have been used in these studies are Eclipse, NetBeans, and Mozilla. Therefore, working on real-world data (or CSPs) and studying the applicability of the machine learning methods in this domain helps us to know if these methods are successful in CSPs or not. This fact encouraged us to study the bug-assignment problem in real-world environments. Also, previous works showed that in recent years different methods ranging from machine to deep learning, text mining and optimization have been used to cope with the bug assignment problem. It should be noted that in this article the emphasis is not on improving machine learning methods or other methods, but the main emphasis is on using these methods in the real world. Hence, some clustering algorithms have been used in their classical form.

Table 1: A survey on previous work, used methods and datasets

Ref.	Method	Dataset	Dataset type
Limsettho et al. (2016) [9]	Topic Modeling, EM, X-Means	HTTPClient, and JCR	OSP
Alenezi et al. (2013) [10]	Naive Bayes	Eclipse-SWT, Eclipse-UI, NetBeans, Maemo	OSP
Florea et al. (2017) [11]	CNN and RNN	Netbeans, Eclipse and Mozilla	OSP
Shokripour et al. (2015) [12]	ABA-Time-tf-idf	Eclipse, NetBeans, ArgoUML	OSP
Shokripour et al. (2012) [13]	IE methods	Eclipse, Mozilla, and Gnome	OSP
Guo et al. (2020) [14]	CNN	Eclipse, Mozilla and NetBeans	OSP
Sajedi-Badashian, and Stroulia (2020) [15]	Vocabulary and Time-aware Bug-Assignment (VTBA)	13 popular GitHub projects	OSP
Zhang, and Lee (2013) [16]	Unigram Model (UM)	Jboss, and Eclipse	OSP
Wu et al. (2011) [17]	KNN, expertise ranking	Mozilla Firefox	OSP
Kashiwa, Y. (2019) [18]	Mathematical programming	Mozilla Firefox, Eclipse, and GNU compiler collection (GCC)	OSP
Goyal, and Sardana (2019) [19]	Bagging, Boosting, Majority Voting, Average Voting, and Stacking	Mozilla Firefox, Open Office, and GNOME	OSP
Yadav et al. (2019) [20]	DES based online system	Mozilla, Eclipse, Netbeans, Firefox, and Freedesktop	OSP
Otoom et al. (2019) [21]	SVM	AspectJ, Tomcat, SWT	OSP
Yadav et al. (2018) [22]	A metric based method	XYZ, pvt. Ltd. India	CSP
Choquette-Choo et al. (2019) [23]	Dual DNN	Google Chromium project	OSP
Zhang et al. (2015) [24]	KSAP	Mozilla, Eclipse, Apache Ant, and ApacheTomcat 6	OSP
Lee, Sun-Ro, et al. (2017) [25]	CNN, Word Embedding	JDT, Platform, Firfox /A,B,C,D	OSP/CSP
Sharma and Singh (2016). [26]	ARM, X-Means	Thunderbird, Add-on SDK, and Bugzilla	OSP
Satish, and Mahendran (2018) [27]	Page ranking and graph databases	QT Framework	OSP
Lee et al. (2019) [28]	GA	Mozilla's Firefox	OSP

Methodology

This section presents the proposed method in detail. The proposed method is aimed to triage the bug report and assign it to an appropriate developer with acceptable speed when a bug is reported in a CSP. The proposed method uses clustering techniques to classify similar bug reports. To select the appropriate clustering algorithm, they are evaluated and the most appropriate algorithm is selected. Determining the similarity between bug reports is done by analyzing their context. Before discussing the proposed method more accurate, some of its advantages are as follows:

- In the real world, we usually face a lot of errors, and it is possible that many of the reported errors are of the same type and go into the fixed state without being checked and no assignment is made to them. The proposed method help the triager to mitigate this problem.
- The speed and accuracy of the bug assigned to the developer increases and developers who have to do the debugging are more accurately identified.
- We can manage all the bugs that affect a specific business or a particular software feature in a single cluster and get enough information from them.
- All bugs in the bug repository are categorized and grouped, making it easy to access and manage as well as reporting.

In software companies, the bug assignment is done by the bug triager manually. She/he checks the reported bugs and select the appropriate developer(s) to fix them. In the proposed method, we use the assignments proposed by the experts as our reference. Hence, the clustering algorithm that suggests the assignments that are closer to the assignments of the experts is preferred.

A. Description of the real-world case study

The proposed method is aimed to process the real data of a web-based system developed (we call it xyzSystem) in a software company. The data used here are the bug reports that have been stored in the BTS of the software company. The BTS maintains the bug reports of three software projects. These projects belong to a larger project. Three software work together to achieve a common business goal. One of the software mentioned as the main software receives online services from the other two software. The purpose of this web-based software is to manage corporate purchases.

B. Data Gathering

The users of the xyzSystem can report the bugs during working with the system. For this purpose, they log in to the ticketing system and send the bug report directly to the BTS. The BTS records are processed by the change control board and after validating the bug report, the appropriate developer is selected to fix the bug. This process is done manually. So, it is a time-consuming task. Automating this task helps the maintenance team to save time and cost and user satisfaction increases.

To apply the proposed method, the bug reports submitted by the users through a ticketing system is used. The bug reports are maintained in the BTS repository. For this purpose, the bug reports are extracted from the BTS using a wrapper, converted to the appropriate format, and arranged in an Excel file. The steps for preparing data can be seen in Fig. 1.

C. Overview of the Method

The steps of the proposed method can be seen in Fig. 1. Bug reports in software bug tracking systems are the information needed to start the proposed method. At the start of the process, the bug reports extracted from the bug tracking system repository are used as the input of the process. Next, the preprocessing step is applied

By applying the preprocessing steps to the bug reports, each bug report is converted to a term vector. After creating the term vector, the similarity between each vector is calculated. By creating the similarity matrix, we are ready to apply clustering algorithms. Finally, we apply tagging on the clusters.

D. Pre-Processing steps

As shown in Fig. 1, the description and summary of each bug report are extracted and used as the input of the preprocessing step. The steps of the pre-processing phase convert raw data into the useful data.

The content of a bug report (which is a bug summary and description) contains information such as time the bug occurred, the location of the bug, and the cause of the bug. At first, the tokenization operation is applied to

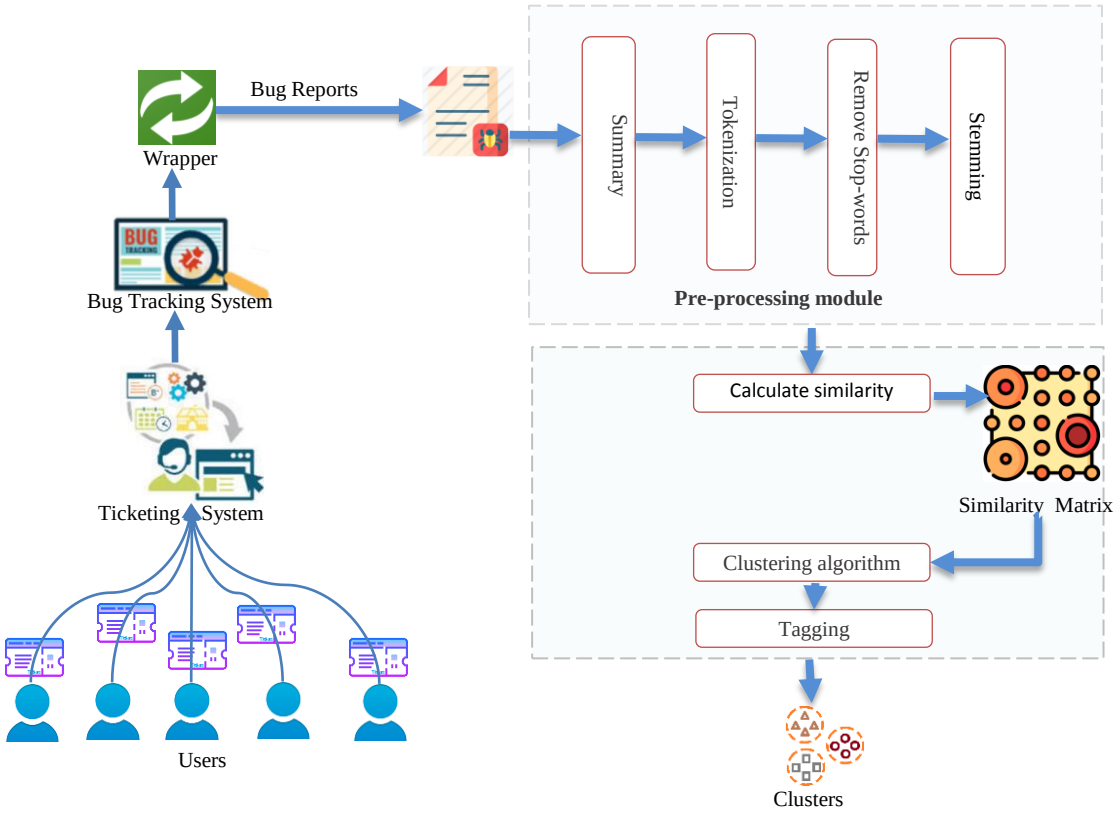

Fig. 1: Schematic diagram of the proposed method.

the extracted text of the bug report. In this way, the textual content of the bug report is converted to tokens. After that, Stop words are removed. The stemming (convert the word to base form) operation is performed on tokens. As an example of the operations in the preprocessing phase, Fig. 2 shows a sample bug report related to the xyzSystem, and the output of the pre-processing steps on the bug reports is shown in Fig. 3.

2019-04-08 10:13:36

Exception in org.xyzSystem.dominant.dao.core.nonPlanAllocation.INonPlanAllocationRepository.getAllGrid()

with cause = 'org.hibernate.exception.SQLGrammarException:

could not extract ResultSet' and exception = 'could not extract ResultSet;

Fig. 2: Summary of a bug report.

2019, 04, 08, 10, 13, 36, exception, org, xyzSystem, dominant, dao, core, nonplanallocation, inonplanallocationrepository, getallgrid, cause, org, hibernate, exception, sqlgrammarexception, could, extract, resultset, exception, could, extract, resultset

---

Fig. 3: Result of pre-processing steps.

E. Calculate Similarity

After applying the pre-processing step to the bug reports, the term vector of each report is calculated. The number of repetitions per term in each bug report is the term vector of that bug report. After calculating the term vector of bug reports, the similarity between the term vectors is calculated using Pearson's correlation coefficient. This is one of the most frequently used methods for calculating the data dependencies [29].

This coefficient is between 1 and -1 and is zero if no relationship exists between the two variables. The formula for calculating Pearson's correlation coefficient is as follows:

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[n\sum x^2 - (\sum x)^2][n\sum y^2 - (\sum y)^2]}} \quad (1)$$

In Pearson's formula, the values of x and y represent two vectors and the value of *n* represents the number of terms involved in calculating the term vector of bug reports. The similarity of each pairs of bug reports is calculated and the similarity matrix is generated between bug reporting vectors. As an example, Table 2 shows the similarity matrix of four typical bug reports in xyzSystem. According to Table 2, the values in the cells of the matrix indicate the similarity between bug reports in the related row and column.

Table 2: Similarity Matrix

	Bug1	Bug2	Bug3	Bug4
Bug1	1	-0.18786	0.099853	0.948872
Bug2	-0.18786	1	0.546667	-0.13032
Bug3	0.099853	0.546667	1	0.099853
Bug4	0.948872	-0.13032	0.099853	1

According to the permissible values of the correlation coefficient, if the similarity value obtained is near to 1 indicates the similarity and if is near to -1 indicates the non-similarity of the bug reports. According to the values in Table 2, reporting bug 1 and 4 with similarity values close to one are more similar, and error reporting 1 and 2 with similarity values near negative are non-similar. If we set the number of clusters to three by default, two bug reports 1 and 4 falls into one cluster and two other bug reports fall into separate clusters.

F. Clustering

The similarity matrix generated in the previous step is used as the input to the clustering algorithms. Clustering algorithms cluster the error reports in the matrix based on their similarity values. The clustering procedure in the proposed method detects related bug reports. Similar bug reports fall into a cluster. The clustering algorithms used here are: 1) EM, 2) Farthest First, 3) Hierarchical Clustering, and 4) Simple K-Means. The clustering algorithms implemented in Weka version 3.6.9 are used here.

G. Tagging

The purpose of clustering is to assign tags to objects that represent each object's membership in the cluster. These tags are keywords that indicate the identity of the content of the cluster.

After clustering, the bug reports should be tagged on the clusters. Usually, the suggestions for selecting a tag can be based on the number of repetitions of the term in the bug reports and the term that is most frequently repeated in the bug reports is selected as the cluster tag. In the proposed method, depending on the clustering issue associated with clustering bug reports and selecting the appropriate developer to fix the bug, each cluster is tagged with the developer specification that has fixed the bug.

Now, when a new bug report occurs in the xyzSystem, the steps of the proposed method are applied on it. First the pre-processing step is performed on the new bug report and finally based on the calculated similarity criteria for the new bug report against the clustered bugs, it is added to the most similar cluster. The new bug is assigned to the developer whose name is tagged on the target cluster. Also, the bug status is changed to “assigned”. The next section presents the test results in detail.

Evaluation and Results

It seems that the proposed method provides an efficient way for automating the bug triage process. In this section, the proposed method is tested with real dataset and the performance of the clustering algorithms is investigated.

A. Dataset

In order to evaluate the proposed method and to understand it more accurately and to verify the validity of the proposed method, experiments were performed on the real dataset of the CSP. The dataset used is the content of bugs that occurred within a given period in the xyzSystem and were fixed by developers with relevant knowledge. The content of the bugs is in a text format. The text file contains a summary of the error description with the exact date and time of the error, and the full address description of the class in which the error occurred, and the reason for the error in the summary of the error description. Fig. 2 shows an example of a brief description of a bug that contains the date and time the bug occurred, the location of the bug, and the cause of the bug. The bug text also contains complete bug descriptions that provide complete information about the bug occurring, and lists the classes inheriting from the original bug class, as well as the list of classes from which the bug class inherits. These items are used in the proposed method to extract the required features for clustering. Bugs that occur at different times on the system are stored in the software bug tracking system and from the time the bug was assigned to the developer until the bug is resolved, the history is stored in the system. All bugs resolved by a developer are considered as items in a cluster and that developer characteristic is tagged on the cluster. As an expert opinion, having a thorough knowledge of the system and the operating process of the system, five clusters were extracted from the bug tracking system. These five clusters containing 100, 50, 50, 50, 50 errors, respectively. So, we have 300 system bugs that were reviewed and resolved by five developers. The bugs are clustered by thoroughly examining the bug text, and each cluster is identified by a developer. That is, on each cluster, the name of the developer that should fix the bugs within that cluster is tagged.

B. Expert Opinion

In this work, we use the tags proposed by the expert triager for each bug report as our reference. According to the expert opinion, the tested dataset is extracted from the bug tracking system, with a total of 300 bugs reported during a specific period. A total of 100 bug fixes have been resolved by one developer, which is considered as a cluster, and the rest of the bugs have been evaluated in four 50-batch clusters by four other developers. Details of the bug reports of the xyzSystem suggested based on the opinions of the expert bug triager are shown in Table 3.

Table 3: Expert opinion specifications about the dataset and tag of each bug report

# of developers	5
# of clusters	5
# items in cluster	50, 100,50,50,50
# bug reports	300

C. Evaluation

After determining the optimal clusters based on the expert opinion, we are ready to evaluate the performance of the clustering methods. The accuracy of the existing clustering algorithms are measured and the algorithm with the near output to the expert opinion is determined. After that, the selected algorithm is used for clustering the new bug reports. Table 4 shows the output of four clustering algorithms. The second column shows the number of clusters. We set this number at 5, because we have 5 active developer in our case study. The third column shows the distribution of bugs in five clusters. For example, in EM algorithm we can see that the first and second cluster contains 49 and 101 bug reports respectively. Each of the three remaining clusters contains 50 bug reports. The fourth column shows the error rate of the corresponding algorithm. The error rate represent the number of bug reports that are incorrectly clustered.

Table 4: Evaluation of the clustering algorithms in terms of error rate and distribution of bug reports in clusters.

Algorithm	# of cluster	Cluster Instances	Error rate
EM	5	49 (16%) 101 (34%) 50 (17%) 50 (17%) 50 (17%)	0.3%
Farthest First	5	49 (16%) 101 (34%) 50 (17%) 50 (17%) 50 (17%)	0.3%
Hierarchical Cluster	5	100 (33%) 100 (33%) 50 (17%) 49 (16%) 1 (0%)	17%
Simple Kmeans	5	7 (2%) 0 (17%) 43 (14%) 101 (34%) 99 (33%)	19%

As can be seen from Table 4, the EM and FarthestFirst algorithms with 3% error rate are the most suitable algorithms for clustering. In EM and Farthest First algorithms, only one bug report is clustered incorrectly. It seems that one bug report from the first cluster is

determined as a member of the second cluster incorrectly. The Hierarchical Cluster and Simple Kmeans algorithms with 17%, and 19%, respectively obtain the next ranks. In hierarchical clustering, most of the bug reports that belong to the fifth cluster are incorrectly assigned to the first cluster. In simple Kmeans, we can see a different behavior where most of the members of the first cluster and all the members of the second cluster are recognized as the members of the fourth and fifth clusters. In general, all the algorithms examined in this work on real data have more that 80% accuracy. However, the EM and FarthestFirst algorithms have similar results, consistent with our expert opinion, and have competitive results. So we can use either of these two algorithms to cluster the bugs. The results of the two Hierarchical Cluster and Simple Kmeans placed at the third and fourth ranks respectively.

D. Applicability and limitations

The results showed that the proposed method based on the clustering techniques has the ability to generate good results for the xyzSystem. It seems that the proposed method is applicable to triage bugs in other CSPs. Usually, similar scenario is used by software companies to receive bug reports in the maintenance phase and triage the reported bugs. In this study, we have extracted 300 bug reports from the BTS to generate clustering models. However, several thousand bug reports available for large-scale OCPs such as Mozilla, Eclipse, etc. that have been used by authors in previous works. Hence, larger datasets in CSPs is recommended to be used in order to study the behavior of the clustering algorithm in such situations. Because the expert opinion is used for measuring the correctness of algorithms, there are factors that may have a negative impact on the algorithm. There must be assurance of the correctness of the expert opinion during different periods of time. Certainly, one of the limitations and challenges will be the inability to confirm the current evolution of expert opinion over time due to changes in the structure of projects and the updates of the development technologies. Another challenge is that past errors may not bear any resemblance to new errors. This is occurred due to possible changes in the project or the organization's focus on new projects and lack of investment in the support and development of past software systems. Another challenge is the change of the structure of the developer teams that can be a weakness in the assignment system because labels on clusters may be the names of developers who are blocked and no longer have a role in developing and supporting systems.

Conclusion

Identifying previously bug reports can reduce the cost of maintaining software. This paper proposed a method for clustering similar bug reports based on the similarity of the contextual content of the reported bugs. For each cluster, the corresponding developer's name is tagged. The calculation of similarity between bug reports is performed using Pearson's correlation coefficient. Four clustering algorithms have been evaluated by the considering the expert opinion. The appropriate algorithm with 3% error is selected for clustering. It seems that the proposed method and similar works can play an important role in maintenance phase to reduce the cost and speed up the bug fixing process. They can be used as an assistance for the bug triager or change control board in software development companies. However, more studies are needed to investigate different aspects of applying automation methods for bug triaging in CSPs.

Author Contributions

M. Yousefi collected the data and designed the experiments. A. Akbari carried out the data analysis. S. M. R. Moosavi and R. Akbari validated the results and wrote the manuscript.

Acknowledgment

The authors would like to thank Computer Engineering and IT Department of Shiraz University of Technology and Computer Science, Engineering, and IT Department of Shiraz University.

Conflict of Interest

The author declares that there is no conflict of interests regarding the publication of this manuscript. In addition, the ethical issues, including plagiarism, informed consent, misconduct, data fabrication and/or falsification, double publication and/or submission, and redundancy have been completely observed by the authors.

Abbreviations

There is no abbreviations.

References

[1] G. Jeong et al., “Improving bug triage with bug tossing graphs,” in Proc. 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering: 111–120, 2009.

[2] J. Anvik, L. Hiew, G. Murphy, “Who should fix this bug?,” in Proc. 28th International Conference on Software Engineering: 361–370, 2006.

[3] D. Cubranic, C. Murphy, “Automatic bug triage using text categorization,” in Proc. Sixteenth International Conference on Software Engineering, Citeseer:92–97, 2004.

[4] H. Hu, H. Zhang, J. Xuan, W. Sun, “Effective bug triage based on historical bug-fix information,” in Proc. 25th International Symposium on Software Reliability Engineering: 122–132, 2014.

[5] J. Anvik, “Automating bug report assignment,” in Proc. 28th International Conference on Software Engineering, ACM: 937–940, 2006.

[6] J. Xuan, H. Jiang, Z. Ren, J. Yan, Z. Luo, “Automatic bug triage using semi-supervised text classification,” in Proc. Intl. Conf. Software Engineering & Knowledge Engineering: 209–214, 2010.

[7] N. Bettenburg, S. Just, A. Schro¨ter, C. Weiss, R. Premraj, T. Zimmermann, “What makes a good bug report?,” in Proc. 16th ACM SIGSOFT International Symposium on Foundations of software Engineering, ACM:308–318, 2008.

[8] A. Goyal, N. Sardana, "Analytical study on bug triaging practices," Jaypee Institute of Information Technology, Department of Computer Science and Engineering, Noida, UP, India, 2020.

[9] N. Limsettho, H. Hata, A. Monden, K. Matsumoto, "Unsupervised bug report categorization using clustering and labeling algorithm," International Journal of Software Engineering and Knowledge Engineering, 26(07): 1027-1053, 2016.

[10] M. Alenezi, M. Kenneth, S. Banitaan, "Efficient bug triaging using text mining journal of software," 8(9): 2185–2190, 2013.

[11] A.-C. Florea, J. Anvik, R. Andonie, "Parallel implementation of a bug report assignment recommender using deep learning," Conference Paper inLecture Notes in Computer Science, 2017.

[12] R. Shokripour, "A time-based approach to automatic bug report assignment," Journal of Systems and Software, 102: 109-122, 2015.

[13] R. Shokripour, Z.M. Kasirun, S. Zamani, J. Anvik, "Automatic bug assignment using information extraction methods," in Proc. International Conference on Advanced Computer Science Application and Technologies (ACSAT): 1-7, 2012.

[14] S. Guo et al., "Developer activity motivated bug triaging: via convolutional neural network," Neural Processing Letters, 51: 2589-2606, 2020.

[15] A. Sajedi-Badashian, E. Stroulia, "Vocabulary and time based bug-assignment: A recommender system for open-source projects," Software: Practice and Experience, 50(8): 1539- 1564, 2020.

[16] T. Zhang, B. Lee, "A hybrid bug triage algorithm for developer recommendation. Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC'13, ACM, NewYork, NY, USA: 1088–1094, 2013.

[17] W. Wu et al. "Drex:developer recommendation with k-nearest-neighbor search and expertise ranking," in Proc. the 2011 18th Asia-Pacific Software Engineering Conference, APSEC: 389, 2011.

[18] Y. Kashiwa, "RAPTOR: Release-aware and prioritized bug-fixing task assignment optimization," in Proc. 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME): 629-633, 2019.

[19] A. Goyal, N. Sardana, "Empirical analysis of ensemble machine learning techniques for bug triaging," in Proc. 2019 Twelfth International Conference on Contemporary Computing (IC3): 1-6, 2019.

[20] A. Yadav , S. Singh, J. Su, "Ranking of Software developers based on expertise score for bug triaging," Information and Software Technology, 112: 1-17, 2019.

[21] A. Otoom et al."Automated classification of software bug reports," in Proc. the 9th International Conference on Information Communication and Management: 17–21, 2019.

[22] A. Yadav, D. Singh, "An information-theoretic approach for bug triaging," 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2018.

[23] C. Choquette-Choo et al. "A multi-label, dual-output deep neural network for automated bug triaging," 18th IEEE International Conference On Machine Learning and Applications (ICMLA), 2019.

[24] W. Zhang, S. Wang , Q. Wang, KSAP:An approach to bug report assignment using KNN search and heterogeneous proximity. Article in Information and Software Technology, 70: 68-84, 2015.

[25] S.-R. Lee, et al., "Applying deep learning based automatic bug triager to industrial projects," ESEC/FSE 2017: in Proc. the 2017 11th Joint Meeting on Foundations of Software Engineering, 926–931, 2017.

[26] M. Sharma, V.B. Singh, "Clustering-based association rule mining for bug assignee prediction," International Journal of Business Intelligence and Data Mining, 11(2): 130-150, 2016.

[27] S. C J, A. Mahendran, "Automated bug assignment in software maintenance using graph databases," International Journal of Intelligent Systems and Applications, 2: 27-36, 2018.

[28] J. Lee, D. Kim, W. Jung, "Cost-Aware clustering of bug reports by using a genetic algorithm," J. Inf. Sci. Eng., 35(1): 175-200, 2019.

[29] M.C. Abounaima et al., "The pearson correlation coefficient applied to compare multi-criteria methods: case the ranking problematic," in Proc. 2020 1st International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET): 1-6, 2020.

Biographies

Mehran Yousefi received his BSc from Isfahan University of Technology. Also, he received his MSc from Shiraz University. His research interests are software engineering, program analysis, software security, reliability of software, AI systems, and Big data. He also has experience in developing software using python, java, php, and groovy.

Reza Akbari has a PhD in software engineering from Shiraz University. Currently, he is an associate professor at department of Computer Engineering and Information Technology of Shiraz University of Technology. His special fields of interest include software engineering in general, machine and deep learning, and optimization algorithms.

Mohammad Reza Moosavi received M.S. and Ph.D. in Software Engineering from Shiraz University, where he is currently an assistant professor. His research interests are Data Mining, Statistical Pattern Recognition and Distributed Systems. He also has teaching experiences especially in field of graph mining, formal methods and distributed systems.

How to cite this paper:
M. Yousefi, R. Akbari S.M.R. Moosavi, "Using machine learning methods for automatic bug assignment to developers," Journal of Electrical and Computer Engineering Innovations, 8(2): 263-272, 2020.

DOI: 10.22061/JECEI.2020.7212.370

URL: http://jecei.sru.ac.ir/article_1471.html