e-Informatica Software Engineering Journal, Volume 11, Issue 1, 2017, pages: 117–141, DOI 10.5277/e-Inf170106

Machine Learning or Information Retrieval Techniques for Bug Triaging: Which is Better?

Anjali Goyal*, Neetu Sardana*
*Jaypee Institute of Information Technology, Noida, India
anjaligoyal19@yahoo.in, neetu.sardana@jiit.ac.in

Abstract

Bugs are the inevitable part of a software system. Nowadays, large software development projects even release beta versions of their products to gather bug reports from users. The collected bug reports are then worked upon by various developers in order to resolve the defects and make the final software product more reliable. The high frequency of incoming bugs makes the bug handling a difficult and time consuming task. Bug assignment is an integral part of bug triaging that aims at the process of assigning a suitable developer for the reported bug who corrects the source code in order to resolve the bug. There are various semi and fully automated techniques to ease the task of bug assignment. This paper presents the current state of the art of various techniques used for bug report assignment. Through exhaustive research, the authors have observed that machine learning and information retrieval based bug assignment approaches are most popular in literature. A deeper investigation has shown that the trend of techniques is taking a shift from machine learning based approaches towards information retrieval based approaches. Therefore, the focus of this work is to find the reason behind the observed drift and thus a comparative analysis is conducted on the bug reports of the Mozilla, Eclipse, Gnome and Open Office projects in the Bugzilla repository. The results of the study show that the information retrieval based technique yields better efficiency in recommending the developers for bug reports.

Keywords: bug triaging, bug report assignment, developer recommendation, machine learning, information retrieval

1. Introduction

The explosive growth in size and scale of software systems has led to the creation of various open source bug tracking repositories. Bug tracking repositories gather, organize and keep track of all the reported bugs. Although, a large number of bug reports help to make the final software product error free, it is really challenging for the bug triager to handle such a large volume of reported bugs. When a new bug is reported, a bug triager analyses the feasibility of bug to verify if the reported bug is not a mere duplicate and contains enough information to be reproduced. If the bug is found to be feasible, it is assigned to a developer for resolution. For effective bug resolution, it is extremely important to assign the reported bug to a suitable developer. Bug assignment is an integral part of bug triaging whose goal is the process of assigning a suitable developer to the reported bug. The assigned developer performs various checks and changes in the source code to rectify the reported issue. The selection of a suitable developer for the bug report is a challenging process as it significantly affects time and cost incurred in the project. Thus, it is imperative to make an appropriate developer assignment who is an expert in the area of the reported bug.

In the past, software projects were small in size and the count of bugs was minimal. In those days, it was possible for the bug triager to perform developer assignment manually but with passing time software projects grew in scale and size. Subsequently, software projects became more complex and in the current scenario, it has

become really cumbersome for the bug triager to be aware of the expertise of all the developers in a triaging team. To ease the task of the bug triager, various semi and fully automated bug assignment approaches have been proposed in the literature. These approaches gather the information related to developer expertise from various sources and utilize it to make developer recommendations. However, the availability of a huge amount of bug assignment approaches appeals for a comprehensive overview.

At present there is no in-depth and focused survey available specifically in the area of bug triaging. It has been observed that only J. Zhang et al. [1] and T. Zhang et al. [2] reported short discussions on bug triaging in their broad category survey on bug handling. This paper performs a systematic, in-depth and focused literature survey on bug triaging. In this paper, 75 papers from peer reviewed, refereed conferences and journals published during years 2004 to 2016 are summarised in an organized manner. The existing approaches are classified into seven categories: machine learning (ML), information retrieval (IR), auction, social network, tossing graphs, fuzzy set and operational research based techniques. The authors further perform an analysis of these approaches in two perspectives: cumulative frequency distribution and year wise trend analysis. In addition, they compare the identified bug triaging techniques inferred from analytical analysis to find the best bug triaging technique.

The rest of this paper is organized as follows: Section 2 presents the anatomy of a bug report and its life cycle. Section 3 describes the systematic survey process. Section 4 reviews the work on bug report assignment and presents a comparative study on two most popular bug assignment techniques. Section 5 concludes this paper and provides some interesting future research directions.

2. Anatomy of a bug report

A bug report is a detailed record constituting a full description related to a bug discovered in any software. It is generally created by the customers, users, developers or testers of software system. A decent bug report ought to comprise three underlying components:

1. Steps to replicate the bug.
2. What is the reporter expected to see?
3. What did the reporter actually see?

A bug report constitutes a collection of various categorical and free form textual data. The categorical data (or meta-fields) constitute fields, such as bug id, product, component, resolution, status, version, priority, creation date, operating system. The free form textual fields contain keywords, summary, description and comments posted by the developers for discussing a probable solution for fixing the bug report. Figure 1 shows an instance of a bug report in the Mozilla project.

Throughout its lifetime, a bug report goes through a number of stages. Various fields, such as status and resolution, vary many a times. When a bug is reported, the status of bug report is marked as New. The triager then assigns the bug to a developer and its status is marked as Assigned. The developer then fixes the issue and the bug is marked as Resolved. If the tester finds the fix to be correct, the bug is marked as Verified and if not, it is Reopened. After the verification of the bug, it is Closed. At the Resolved status, there are multiple resolutions such as Fixed, Duplicate, Won't Fix, Non-reproducible and Invalid. Figure 2 shows the basic life cycle of a bug report.

3. Systematic review process

This section presents the survey process used in this work. The guidelines of the systematic literature review (SLR) by Kitchenham and Charters [3] were used in this work.

3.1. Survey process

The review process was started with an initial search where the renowned journals and conference proceedings which contained papers concerning bug triaging were selected. Other used materials encompassed even e-sources relevant to

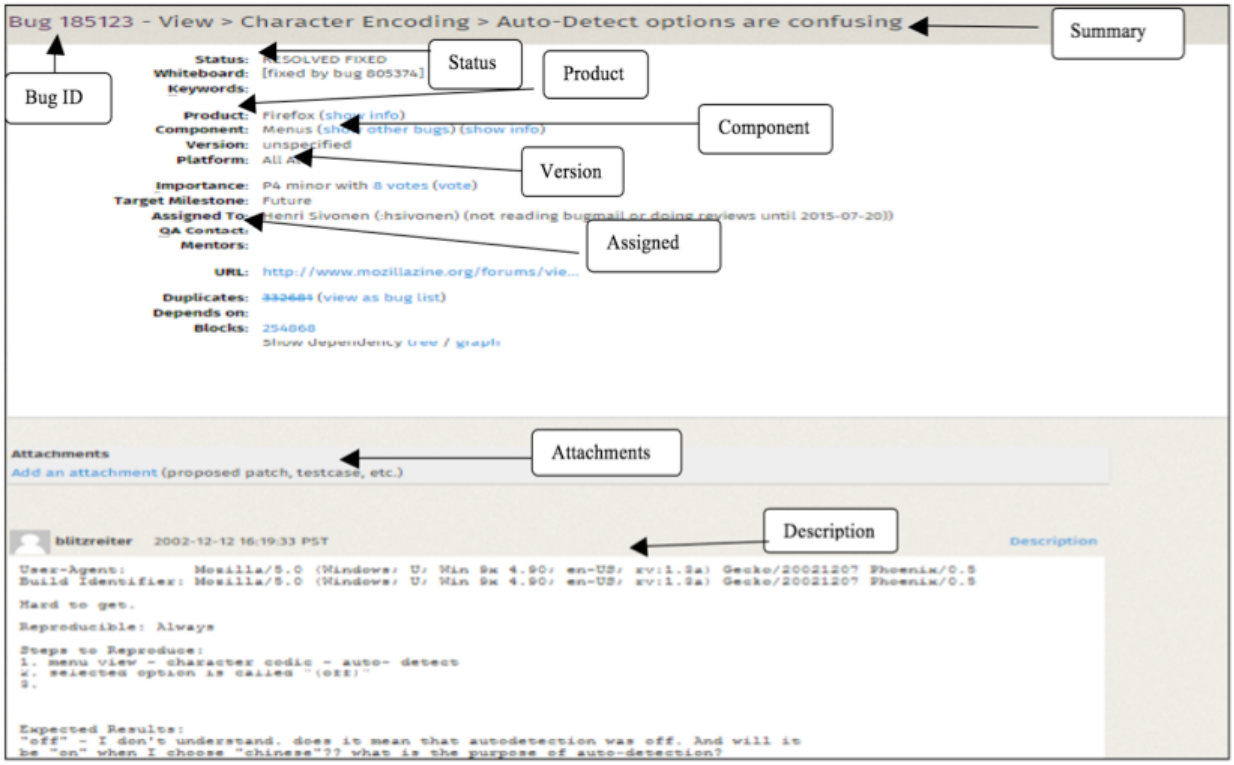


Figure 1. An instance of bug report

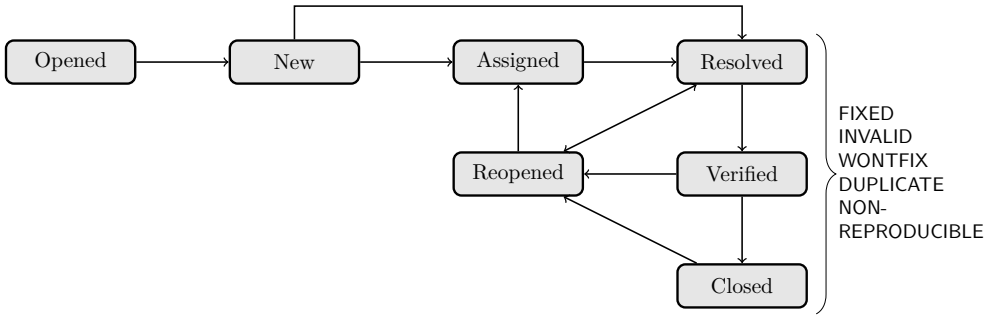


Figure 2. Life cycle of a bug report

software engineering: IEEExplore, ACM Digital library, Google scholar, Citeseer library, Inspec, ScienceDirect and EI Compendex. Selecting such venues ensured that the selected articles meet worthy standards.

To further ensure that no important papers in bug assignment are missed, certain keywords closely related to bug report assignment were identified in the articles obtained from the above venues. A google search was performed to find the identified keywords: bug triaging, bug fixing, bug resolution, bug report assignment and bug AND developer recommendation. These keywords were intentionally broad enough to cover as many articles as possible, although many were less relevant to the present scope of the study. After performing the preliminary keywords and venue search, the studies that propose new bug assignment algorithms were identified. A large number of papers in the keyword search also resulted in papers other than bug report assignment, such as bug duplication, bug localization, severity/priority prediction. All such papers were excluded from this review. After reviewing the titles, abstracts and

Table 1. Distribution of reviewed papers among various sources

Type	Acronym	Description	No. of papers
Journal	JSEP	Journal of Software: Evolution and Process	3
	JSS	Journal on Systems and Software	2
	JSW	Journal of Software	2
	TSE	IEEE Transaction on Software Engineering	2
	Others		**7**
	Total		16
Conference	APSEC	Asia Pacific Software Engineering Conference	2
	ESESC/FSE	European Software Engineering Conference/ ACM SIGSOFT Symposium on the Foundations of Software Engineering	3
	ICSEA	International Conference on Software Engineering Advances	2
	ICSE	International Conference on Software Engineering	5
	ICPC	International Conference on Program Comprehension	2
	ICSM	International Conference on Software Maintenance	3
	ICT-KE	International Conference on ICT and Knowledge Engineering	2
	MSR	Mining Software Repository	7
	PROMISE	International Conference on Predictive Models in Software Engineering	2
	SAC	ACM Symposium on Applied Computing	3
	SEKE	International Conference on Software Engineering and Knowledge Engineering	2
	ESEM	International Symposium on Empirical Software Engineering and Measurement	3
	COMPSAC	International Conference on Computers, Software and Applications	2
	Others		**21**
	Total		59
TOTAL PAPERS (16 + 59)			75

skimming through full articles wherever required, finally 75 papers were reviewed in this study. Each paper was thoroughly verified to assure its the correctness and relevance. Table 1 enlists the distribution of papers across various sources concerning bug report assignment. The venues at which only one surveyed paper was published are grouped together in the "Others" category.

3.2. Inclusion and exclusion criteria

This paper surveys the articles meeting the following inclusion and exclusion criteria:

Inclusion criteria:

1. Papers must relate to developer assignment in bug repositories.
2. Papers must describe the methodology and experimental evaluation of proposed algorithms.
3. Papers must be published in peer reviewed journals and conferences.

Exclusion Criteria:

1. Papers that are duplicates of similar work.
2. Papers that do not describe the methodology and experimental evaluation.
3. Papers that are not published in peer reviewed venues.

3.3. Related surveys

In the past, J. Zhang et al. [1] and T. Zhang et al. [2] performed surveys closely related to this work. These surveys cover all the stages of bug handling, i.e. bug report analysis, bug triaging and bug fixing as shown in Figure 3. Short discussions related to all these stages were carried out in their respective studies. However, a comprehensive overview on each individual stage of bug handling is still missing. This paper focuses on the second stage of bug handling, i.e. bug triaging (or bug report assignment). Bug triaging is an integral stage of bug handling which focuses on the selection of a suitable developer for bug fixing. Hence, this work presents the first large-scale, in-depth, and focused study of bug report assignment. J. Zhang et al. [1] reviewed

Figure 3. Classification scheme for bug handling process

14 papers whereas T. Zhang et al. [2] reviewed 21 papers related to bug report assignment in their respective studies. The range of surveyed papers covered in this work is larger than in these earlier works. This investigation encompassed the reviews of 75 papers on bug report assignment. It covers papers published before July 2016.. Hence, this study is more comprehensive and up-to-date as compared to the other surveys.

3.4. Research contribution

The following new research contributions differentiate this work from the prior studies:

1. This paper presents the first in-depth, systematic and focused survey on bug triaging considering 75 papers from peer reviewed, refereed conferences and journals published during years 2004 to 2016.
2. Inference drawn from the systematic literature review illustrates ML and IR to be the most popular bug triaging techniques. Thus, a comparison of these popular techniques was done to identify the best bug triaging technique.
3. The paper presents the experimental results of the empirical analysis of two four scale open source projects, Mozilla, Eclipse, Gnome and Open Office of the Bugzilla repository.

4. Bug report assignment

Numerous researchers proposed different bug assignment approaches to semi or fully automate the developer recommendation process. Bug assignment approaches can be classified by the methodology used in the recommendation process. It can be divided into two broad categories: activity profiling of developers [4–8] and location based techniques [9, 10]. The general idea behind the activity profile based techniques is to develop an expertise profile of each developer by using topic modelling. A list of topics is made on the basis of historically fixed bug reports and a membership score is computed for each developer with respect to each topic. This score represents the involvement of a developer in a particular topic in the past. For any new bug report, the topics are extracted and the developer with maximum score corresponding to the obtained topics is recommended. The activity profiling of developers suffers from two major problems: a) Obsolete profiles after some time, b) The developers switch teams or new developers are added, which changes the developer profiles to a major extent thus reducing the recommendation accuracy after some time. However, the high efficiency achieved by activity profile based approaches when the profiles are updated cannot be overlooked.

The location based bug triaging techniques, on the other hand, locate the source code files that need to be updated in order to resolve the issue. The developers who had earlier worked upon these files are considered to be suitable for further updating of these files. These approaches usually make use of the version control repository of the project and thus the data source is more reliable. However, the two-level predictions, firstly the source code files that need to be changed in

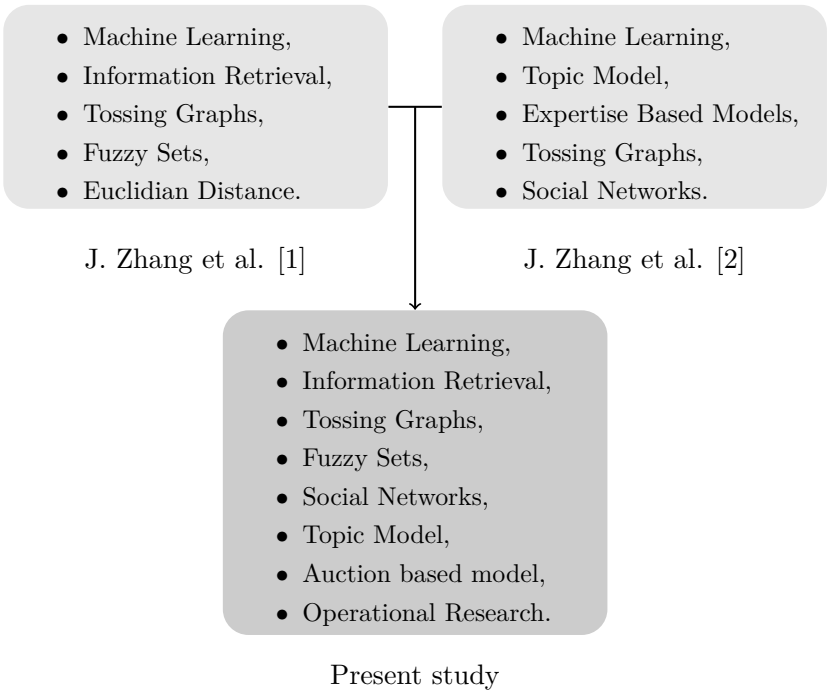


Figure 4. Classification categories in different studies

order to fix the bug and secondly the developer choice, limit the accuracy of the location based approaches as compared to the activity profile based approaches. Therefore, the activity profile based approaches are more popular for bug report assignment in industry.

4.1. Classification based on popular techniques

J. Zhang et al. [1] conducted their study in 2015 and identified bug triaging into five categories: machine learning, information retrieval, tossing graphs, fuzzy set and the Euclidean distance. T. Zhang et al. [2] conducted their study in 2016 and identified bug triaging into the categories: machine learning, topic model, tossing graphs, social networks and expertise model based techniques. In this study, seven categories for bug report assignment were identified after careful inspection. The categories considered first are the ones which were present in both studies, i.e. machine learning and tossing graphs. Next, the papers related to topic modelling and the expertise based model in the category information retrieval were added. The categories which are present in either of the previous studies, i.e. fuzzy sets and social networks, were also considered. In addition, two new categories were identified: auction based techniques and operational research (OR) based approaches. Further, the OR based category include the work related to the areas: Euclidean distance, genetic algorithm and greedy optimization. Hence, after a systematic evaluation of literature, finally seven categories for bug report assignment were inferred: machine learning, information retrieval, auction, social network, tossing graphs, fuzzy set and operational research based techniques. Figure 4 shows the classification categories considered in different studies. Among the seven identified categories, machine learning and information retrieval based techniques are automated and the others are semi-automated. The literature was classified and reviewed under these seven heads as follows: **Machine learning based approaches** (see Tab. 2). These approaches train a supervised or unsupervised machine learning classifier with bug reports fixed in the past and then use it for the selection of prominent developers for new bug reports. Cubranic et al. [11] presented one of the few initial bug report assignment approaches based on

supervised machine learning classification. They considered the report developer assignment as a text classification problem and trained the machine learning classifier using the tokens obtained from a textual description of fixed bug reports. They correctly classified 30% of Eclipse bug report assignments using supervised Bayesian learning. Xuan et al. [12] highlighted a drawback resulting from the deficiency of labelled bug reports in bug repositories. They first labelled all the unlabelled bug reports using a combination of Naïve Bayes and the Expectation Maximization algorithms and then used the labelled data for training machine learning classifiers.

Anvik et al. [13] recommended to use eight information sources for developer assignment in contrast to the usage of tokens obtained only from a textual description of bug reports. They proposed to use the textual description, component, operating system, hardware, version, developer who owns the code, current workload of developers and developers actively participating in the project to select a prominent developer for a new bug report. They classified bug reports using the support vector machine algorithm and obtained 57% precision for the Eclipse project and 64% precision for the Mozilla project [14]. Although, the inclusion of eight information sources augmented the proficiency of bug assignment selection, sometimes it is still not probable that all the designated eight parameters from each bug tracking system will be obtained.

For instance, large open source bug repositories do not distribute the data concerning the workload of their developers. Thus, it is not always practical to incorporate all the eight fields. Anvik et al. [15, 16] extended their work in order to equate various machine learning classifiers, such as Naïve Bayes, Support Vector Machine (SVM), C4.5, Expectation Maximization, Conjunctive Rules and the Nearest Neighbour (NN) algorithm. Their experimental results exhibited that SVM is the most efficient tool for bug assignment. Similarly, Lucca et al. [17] compared k-NN, SVM and the probabilistic model for bug report assignment.

Bhattacharya et al. [18] surveyed the influence of different dimensions on bug report assignment. They studied how different dimensions such as the choice of a classifier, feature selection, the inclusion of tossing graphs and incremental learning affects bug triaging. Their investigation showed that the Naïve Bayes classifier and the product-component pair as the parameters and the inclusion of tossing graphs along with incremental learning are the best suited dimensions for bug triaging. Their approach achieved significant reduction in tossing lengths. Hu et al. [19] presented a developer-component-bug based bug triaging framework, BugFixer. Xuan et al. [20] focused on the problem of using large datasets for bug assignment, thereby increasing the computation time and complexity of different algorithms. They utilized the combination of feature and instance selection algorithms to choose a dataset for the training of a classifier. Their results demonstrated that scaling down the dataset significantly diminishes the computation complexity and also increases the classification accuracy. Xia et al. [21] proposed DevRec, a dual analysis model which consists of bug report (BR based) and developer (D based) analysis. DevRec is tested on five large projects: GNU, Compiler Collection, Open Office, Mozilla, NetBeans and Eclipse. The precision@5 and precision@10 of DevRec vary from 21.00% to 31.96% and 13.31% to 18.59%, respectively [22].

Machine Learning based approaches consider bug report assignment as a single-label learning problem. In the previous studies, Naïve Bayes is the most popular classifier in machine learning based approaches and it is extensively experimented on in the bug reports of the Bugzilla repository.

Information retrieval based approaches (see Tab. 3). These approaches consider bug reports as documents and transform them to feature vectors which are then processed for optimal developer assignment. These approaches work on the principle that developers with similar expertise towards a certain kind of bugs are proficient enough to solve the new bug report of a similar kind. These techniques consider developer's past expertise towards historically fixed bug reports so as to select a prominent developer.

Moin et al. [23] presented an n-gram based string matching algorithm for bug triaging in the Eclipse JDT project. They transformed the historically fixed bug reports to n-gram tokens. The proposed an approach which matches the n-grams of a new bug report to the n-grams of historically fixed bug reports and allows to find the related fixed bug report. The developer who had fixed the historically similar bug report is designated for the new bug report as well. Matter et al. [4] utilized vocabulary obtained from the source code contributions of developers to build a term-author matrix. Each entry in the matrix represents the frequency of term with respect to a developer. This frequency is considered the expertise of a particular developer with respect to a particular term. For a new bug report, the vocabulary obtained from the textual description of new bug report is matched with the vocabulary of the term-author-matrix and a developer with the highest expertise is designated for the new bug report. Similarly, other researchers used the smoothed unigram model [24], latent semantic indexing [7,25,26], similarity computation [27,28], vector space modelling [28–30] and topic or term modelling approaches [5, 6, 31–37] for developer recommendation. Ahsan et al. [25] implemented dimensionality reduction using feature selection and latent semantic indexing in the expertise matrix.

Somasundaram et al. [38] merged information retrieval with a machine learning based technique for effective developer recommendation. They reviewed three algorithms, SVM-TF-IDF (Support Vector Machine–Term Frequency–Inverse Document Frequency), SVM-LDA (Latent Dirichlet Allocation) and LDA-KL (Kullback Leibler Divergence) and determined LDA-KL to be most effective for developer selection. Shokripur et al. [39] mined information from the version control repository of the project to propose a location based technique for bug triaging. Unlike other approaches, they did not utilize the information obtained from bug tracking systems. Their approach allowed data to be used in new projects also as the underlying data used for recommendation which does not get obsolete after some time.

Shokripur et al. [9] used only the index of unigram noun terms for bug triaging. They concluded that using only unigram noun terms shortens the token index and does not affect the recommendation accuracy. They associated the noun terms with the source code files of the project and then fetched developers who had earlier worked on the linked files for recommendation. Time based expertise decay is also efficient for developer selection in bug report assignment [40–43]. The knowledge of a developer degrades with time. Hence, the calculation of developers' expertise should also comprise time usage as a factor for frequency normalization. This normalization lowers the weight for terms that were previously used and keeps the training data updated. This capability of information retrieval based techniques makes them popular for optimal bug report assignment.

The information retrieval based approaches consider the developer's expertise for bug report assignment. They utilize a large number of the meta-fields of bug reports along with tokens obtained from textual contents. Term frequency modelling is the most popular IR based bug assignment approach.

Auction based approaches (see: Tab. 4). Hosseini et al. [44] proposed an auction based technique for developer recommendation in bug repositories. Upon receipt of a new bug report, the bug triager auctions off the bug report to developers. The software developers who want to work on the auctioned bug report bid to gain it. The bug report is assigned to one of the interested developers on the basis of their bids and current workload status. These techniques are advantageous as the chances of success in such approaches are high as the bidders themselves desire to take the responsibility for fixing the bug. Such approaches usually benefit by moving the style from 'doing the job right' to 'doing the right job'. However, they usually suffers from time delays as the time required for suitable developer assignment is long.

The auction based approach leads to a slower developer assignment process. However, this increases developer's confidence towards a bug.

Social network based approaches (see Tab. 5). A social network refers to the network of social interactions and personal relationships. Social network approaches utilize the relationships between developers and bug reports for the selection of a suitable developer. They compute the developer expertise based on various influencing factors of the network. Various past studies used social network based approaches for bug report assignment [45–49].

The social network based approaches are a recent development in bug triaging. They consider various parameters for decision making and additionally incorporate complex computations in bug triaging task.

Tossing graph based approaches (see Tab. 6). In a normal scenario, a bug triager assigns a bug report to a software developer who makes source code changes in order to fix the bug. If the assigned developer is not able to resolve the bug, then a new developer is assigned for the bug report. Such a process of switching over the bug to new developers is known as bug tossing. Bug tossing is a major problem in bug triaging as approximately 93% of bug reports are tossed at least once in their lifetime [50]. Various researchers propose the use of tossing graph based approaches for bug report assignment in the literature [50–52]. These approaches consider the historical tossing chains illustrating the switching of developers in the past. For a new bug report, the developer is selected on the basis of previous expertise and then tossing chains are checked to identify the most suitable developer.

The tossing graph based approaches help to significantly reduce tossing path lengths. They use various bug meta-fields and topics obtained from textual parameters for similarity calculation and then use historical tossing chains to find the most suitable developer.

Fuzzy set based approaches (see Tab. 7). Fuzzy sets are sets whose elements have degrees of membership. Fuzzy set based bug triaging approaches compute the expertise (or membership score) of developers with respect to various topics obtained from bug parameters. Tamrawi et al. [53, 54] proposed the fuzzy set based approaches in the past. These approaches formulate the term frequency values of IR based approaches into fuzzy set memberships. When a new bug report arrives, matching tokens are obtained and the corresponding membership scores are aggregated.

The fuzzy set based approaches use the fuzzy set theory in which the most descriptive terms characterizing each developer are collected and then used to measure the suitability of a developer for a new bug report.

Operational research based approaches (see Tab. 8). Bug report assignment is an NP hard problem. Hence, different practitioners and researchers have used mathematical techniques, such as greedy optimization, genetic algorithm, the Euclidean distance, to resolve the problem of bug report assignment. Niknafs et al. [55] presented a study on using the genetic algorithm and the multi criteria decision making technique in the personnel assignment problem. Rahman et al. [56] proposed the usage of the greedy optimization technique for developer assignment in bug tracking systems. Xia et al. [21] proposed a machine learning based approach for bug assignment where the similarity between developer and bug report is calculated using the Euclidean distance. Panagiotou et al. [57] proposed the STARDOM approach for bug report assignment and concluded that the analytic hierarchy process (AHP) should be used for the profile construction of developers and for the ranking of developers.

The operational research based approaches utilize mathematical models for bug report assignment. However, scalability issues in such models for large scale open source projects are still questionable.

4.2. Key observations

In this work, the authors have reviewed 75 research papers published during the years 2004-2016. As a result seven categories of bug assignment approaches have been identified. The categories, as mentioned earlier, are: machine learning, information retrieval, auction based, social network, tossing graphs, fuzzy set and operational research based techniques. Based on this in-depth survey, this study is analysed from

Table 2. Comparison of machine learning based approaches

Paper	Year	Technique used	Experimental dataset	Parameters (#Parameters)	Method summary and merits	Remarks/demerits
Cubranic et al. [11]	2004	Machine learning (Naïve Bayes)	Eclipse	Summary and description (2)	Tokens generated from the textual parameters for classification are considered.	30.5% accuracy for eclipse bug reports
Xuan et al. [12]	2010	Machine learning (Naïve Bayes, expectation maximization	Eclipse	Summary and description (2)	The problem of the deficiency of labelled bug reports is considered. Classification efficiency improved by 6% as compared to using only the Naïve Bayes classifier.	Additional overhead to calculate the label of a bug report first.
Bhattacharya et al. [18]	2012	Machine learning (Naïve Bayes, Bayesian Network, J48, SVM)	Mozilla and Eclipse	Product, component (2)	It examined the impact of various dimensions such as classifier selection, feature selection, inclusion of tossing graphs and incremental learning on bug triaging. The Naïve Bayes algorithm is concluded for classifier, product-component pair for features and the use of tossing graphs with incremental learning as the best suited model for bug assignment. Significantly reduced tossing length.	High computational time and cost
Zou et al. [70]	2011	Machine learning (feature selection, instance selection, Naïve Bayes)	Eclipse	Summary and description (2)	Feature selection is used for removing noisy data. It removed 50% of bugs after training set reduction. The performance of original Naïve Bayes algorithm is improved by up to 5% for the Eclipse project.	It used only the Eclipse bug for experimentation. It needs to be applied on more datasetsand also needs more features.
Lin et al. [71]	2009	Machine learning (J48)	Proprietary Chinese dataset	Summary, description, step, bug type, bug class, phase Id, submitter, module Id, bug priority (9)	It proposed a bug assignment model for the Chinese bug dataset. It demonstrated that the non- text based approach outperformed the text based approach.	The proprietary Chinese dataset is used for evaluation. It needs more features.

Banitaan et al. [72]	2013	Machine learning (Naïve Bayes)	Mozilla, Eclipse, NetBeans, Free Desktop	Summary, reporter, component (3)	It demonstrated that the component is the most influential parameter in bug assignment.	It needs more evaluation parameters.
Anvik et al. [73]	2006	Machine learning (Naïve Bayes, SVM, J48)	Mozilla, Eclipse, gcc	Summary and description (2)	The obtained precision was 57% for the Mozilla and 64% for the Eclipse project.	Test set size is too small. 170 for Mozilla and 22 for Eclipse.
Xia et al. [21]	2013	Machine learning (k-Nearest Neighbour)	Mozilla, Eclipse, Netbeans, Open Office, gcc	Product, component, summary, description , developer Id (5)	It proposed Devrec, a composite model which performs two types of analysis: developer based (to find a set of prominent developers) and bug report based analysis (to find similar bug reports).	It needs more features for bug similarity calculation.
Sharma et al. [74]	2015	Machine Learning (association rule mining)	Thunder Bird, Addon SDK, Mozilla	Severity, Priority, Summary (3)	It proposed a novel association rule mining based on bug triaging algorithm.	Reported accuracy up to 81% for top-3 recommendations. More rigorous testing on large dataset required.

Table 3. Comparison of information retrieval based approaches

Paper	Year	Technique used	Experimental dataset	Parameters (#Parameters)	Method summary and merits	Remarks/demerits
Woo et al. [75]	2011	Information retrieval (topic modelling)	Apache, Eclipse, Linux Kernel, Mozilla	Version, platform, milestone, textual description (4)	A content boosted collaborative filtering model which reduces cost without significantly sacrificing accuracy is proposed.	Important bug features, such component in parameter selection, are missing.

Matter et al. [4]	2009	Information retrieval (vocabulary expertise based model)	Eclipse	A developer owns the associated code and description, active participation of developers (3)	The term author matrix to model the expertise of developers was coined here. No need to train the data. The used time decay factor for developers is 3 months.	Low precision value (33.6% for top-1 developer list size in the Eclipse Project)
Shokripour et al. [9]	2013	Information retrieval (location based technique)	Mozilla and Eclipse	Noun terms extracted from source code files, comment messages and identifiers (1)	It proposed a two-phased location based technique for bug report assignment. The first phase predicts the source code files to be updated and second phase lists the probable developers. Reported accuracy up to 89.41% for the Eclipse and 59.76% for Mozilla project for top-5 recommendation list.	Evaluation datasets too small. Test set size:100 No. of unique developers in dataset: 9 and 57 only
Shokripour et al. [40]	2015	Information retrieval (term frequency)	Eclipse, NetBeans, ArgoUML	Noun terms (1)	It proposed a time-based approach term weighting approach. Experimental evaluation shows accuracy improvement up to 11.8%.	No. of unique developer in the dataset too small
Naguib et al. [6]	2013	Information retrieval (term frequency)	Atlas Reconstruction, Eclipse BIRT, UNICASE	Component and description (2)	Proposed an activity profile based technique for bug report assignment.	Considered small datasets for experimentation
Xie et al. [31]	2012	Information retrieval (topic modelling)	Mozilla and Eclipse JDT	Frequent topics in bug report (1)	In the proposed approach Dretom models the developer's expertise based on topic models built from historical fixed bug reports.	Requires more features.
Alenezi et al. [32]	2013	Information retrieval (topic modelling)	Eclipse, NetBeans, Maemo	Bug ID, Assignee, Opened, Changed, Summary, Component (6)	Investigated the use of four term selection methods, namely Log Odds Ratio, Chi-Square, Term Frequency Relevance Frequency and mutual information.	Chi-Square was found to be the best technique.
Alijarah et al. [7]	2011	Information retrieval (latent semantic analysis)	Eclipse	Textual description (1)	It modelled a bug term matrix to compute similar bug reports. The developers are then assigned according to the past expertise.	More rigorous testing is needed.

Anjali et al. [41]	2016	Information retrieval (term frequency)	Mozilla and Eclipse	Component, severity, priority, operating system (4)	It proposed a time decay based technique for bug triaging which uses bug meta-fields to create a term author matrix. The frequency values in the matrix are then degraded according to the last usage time.	More features are needed.

Table 4. Auction based approach

Paper	Year	Technique used	Experimental dataset	Parameters (#Parameters)	Method summary and merits	Remarks/demerits
Hosseini et al. [44]	2012	Auction based technique	Mozilla and Eclipse	Bug id, creation date, last updated date, classification id, product, component, version, platform, operating system, bug status, resolution, duplicate id, bug file location, keywords, priority, bug severity, target milestone, dependent bugs, blocked, votes, reporter name, assigned to name and number of comments (22)	The developers place their own bids as per their own interest/expertise. It reduces the chances of bug tossing. Overall the method saves bug rectification time.	Slow developer assignment process. Low Accuracy values up to 33.54% and 25.14% obtained for the Mozilla and Eclipse projects, respectively.

Table 5. Comparison of social network based approaches

Paper	Year	Technique used	Experimental dataset	Parameters (#Parameters)	Method summary and merits	Remarks/demerits
Wu et al. [45]	2011	Social network	Mozilla	Summary and description (2)	It finds historical similar bug reports with the help of the k-Nearest Neighbour search and then uses in-degree, out-degree, page rank, betweenness and closeness metrics to rank developers from a social network. The obtained recall value is up to 0.60 for the Mozilla project.	Additional cost to adjust algorithmic parameters.

Zhang et al. [46]	2012	Social network	JBoss	Summary and description (2)	It builds the concept profile with the topic terms of bug reports and then uses the social network of developers to find the developer with the highest probability of fixing.	Additional cost to maintain social network of developers. Computationally intensive.
Xuan et al. [47]	2012	Social network	Mozilla and Eclipse	Bug ID, reporter, fixer, summary, description, creation time, and comments (7)	It models the bug report assignment as the developer prioritization problem by extending a socio-technical approach. In the proposed approach, the out-degree of developers is used to construct a social network which is then used in prioritization.	Low accuracy values (up to 50%).
Yang et al. [48]	2014	Social network	JBoss	Bug ID, description, number of comments, summaries, developer ID, component (6)	It proposed a multi developer network based approach for bug report assignment. Potential contributors are extracted from similar component and keyword matching.	Additional bug parameters should be used for similarity matching.
Zhang et al. [49]	2013	Social network	Mozilla and Eclipse	It utilized objects such as developers, bugs, comments, and components, as well as links denoting different relations among these objects. (2)	It introduced a heterogeneous developer contribution network to model the multiple types of contribution from developers to components in software bug repositories.	Computationally complex process.

Table 6. Comparison of tossing graph based approaches

Paper	Year	Technique used	Experimental dataset	Parameters (#Parameters)	Method summary and merits	Remarks/demerits
Bhattacharya et al. [50]	2010	Tossing graph	Mozilla and Eclipse	Bug Id, developer Id, product, component, keywords obtained from summary and description (6)	It utilized multi-feature tossing graphs along with machine learning classifiers to reduce the tossing path lengths and improve the efficiency of a bug recommender. Reduced tossing length up to 86%	More bug parameters should be added for the construction of tossing graphs.

Jeong et al. [51]	2009	Tossing graph	Mozilla and Eclipse	Tossing information of developers (1)	It introduced the use of Markov chains based on tossing graphs for efficient developer recommendation. Reduced tossing events by up to 72%.	Additional cost to obtain the path from the first recommendation to the final fixer.
Chen et al. [52]	2011	Tossing graph	Mozilla and Eclipse	Tossing information of developers, reporter, classification, component, product and summary (6)	It introduced an approach to improve bug assignment using a bug tossing graph and bug similarity (vector space model).	Better bug similarity techniques could be employed.

Table 7. Comparison of fuzzy set based approaches

Paper	Year	Technique used	Experimental dataset	Parameters (#Parameters)	Method summary and merits	Remarks/demerits
Tamrawi et al. [53]	2011	Fuzzy sets	Eclipse	Bug ID, developer ID, summary, description (4)	It introduced the fuzzy set based model for developer recommendation by using membership functions.	Low accuracy: 37.81%
Tamrawi et al. [54]	2011	Fuzzy sets	Mozilla, Eclipse, Apache, NetBeans, FreeDesktop, Gcc and Jazz	Bug ID, developer ID, summary, description, creation/fixing time (5)	It developed the fuzzy set and cache based approach, Bugzie for the calculation of developer expertise. Lower response time and better accuracy as compared to the SVM based classification.	Additional overhead in topic modelling and calculating fuzzy numbers.

Table 8. Comparison of operational research based approaches

Paper	Year	Technique used	Experimental dataset	Parameters (#Parameters)	Method summary and merits	Remarks/demerits
Nikanfs et al. [55]	2010	Operational research based technique (genetic algorithm)	Eclipse JDT and two industrial projects	Current workload of developers (1)	Proposed the genetic algorithm based technique for developer recommendation.	Requires rigorous experimental evaluation.
Rahman et al. [56]	2011	Operational research based technique (greedy optimization)	Industrial project	Fluency, contribution, effectiveness and receny (4)	It proposed a profile creation and maintenance module. The importance factor of developers is calculated based on various features which are further prioritized on the basis of various properties.	Difficult to implement due to the calculation of various extracted parameters.
Karim et al. [22]	2016	Operational research based technique (genetic algorithm)	Eclipse	Component, no. of file changes, fix time. (3)	It proposed a single objective (minimization of bug fix time) and as a bi-objective (minimization of bug fix time and cost). The developer assignment algorithm uses the genetic algorithm.	Two level prediction (fix time and prominent developer) minimizes the accuracy.
Rahman et al. [76]	2009	Operational research based technique (greedy optimization)	Eclipse	Lines of code (1)	It proposed the greedy optimization algorithm for bug report assignment.	Line of code is a weak metric for any kind of prediction as the coding styles of developers vary to a great extent.

two perspectives: the frequency wise distribution of techniques and the year wise distribution of each technique.

- **Frequency wise distribution of each bug assignment technique:** In this perspective, the popularity of all bug triaging techniques identified in this study was analysed. The frequency of each technique was accumulated and the cumulative frequency distribution was developed. The resultant histogram is presented in Figure 5b. A similar analysis was performed on the papers from the existing surveys [1, 2]. The frequency distribution of techniques in their studies is shown in Figure 5a. The conducted analyses show, Figure 5a and 5b, that the machine learning and information retrieval based techniques are most popular for bug assignment.

Another analysis was done in order to check the year wise trend of the bug assignment techniques in the last two decades.

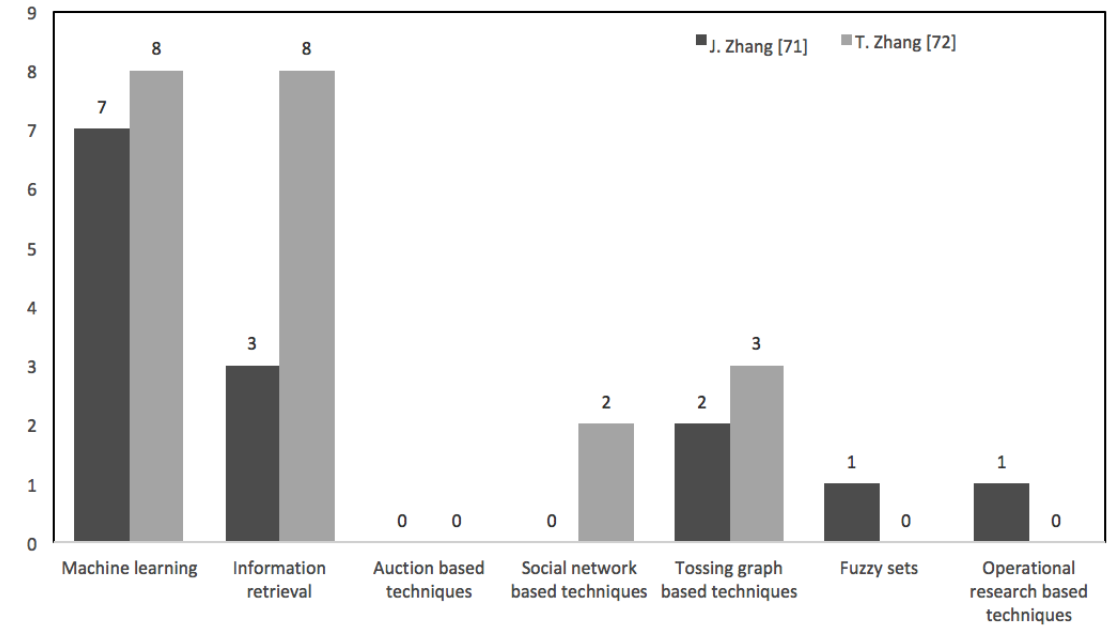
- **Year wise distribution of each bug assignment technique:** a) From the above analysis, ML and IR were identified to be the most popular techniques among all categories of bug triaging. To further analyse the on-going trend among the popular techniques, the year wise frequency distribution of ML and IR based techniques was plotted. Figure 6a shows the year wise frequency distribution of ML and IR based techniques in the existing study [2]. Since, J. Zhang et al. [1] surveyed very few papers on bug triaging, the trend analysis will not give any significant insights. Hence, it is believed that this survey is not useful for this analysis. Figure 6b represents the year wise trend analysis of ML and IR based techniques in the current study. It was observed that there is a considerable trend shift from ML to IR. Researchers prefer to use the IR based technique for bug triaging.

To further examine the reason behind this trend shift, an empirical study on two most popular techniques, ML and IR for bug triaging, was performed.

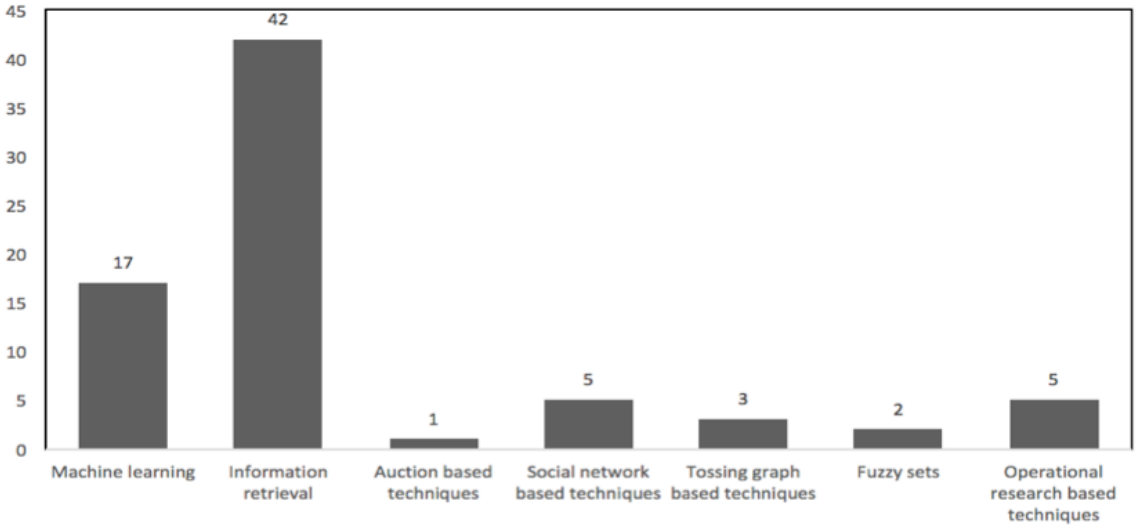
4.3. Comparative study of machine learning and information retrieval techniques

To evaluate the efficiency of the machine learning and information retrieval based techniques, the techniques based on the bug reports of four large scale popular projects of Bugzilla repository, Mozilla, Eclipse, Gnome and Open Office, were applied. Bugzilla is the most popular open source bug repository used by many varied size software projects. The projects selected for the comparative study contain large number of bug reports and are widely used in Bugzilla. They have been developed for years and thus now they are aged. This increases the confidence of researchers in the use of these projects for experimental evaluations in their work.

The datasets for the comparative study were collected from the issue tracking system (ITS) of the Mozilla, Eclipse, Gnome and Open Office projects. Bug reports submitted over the span of 6 years (from January 01, 2011 to December 31, 2016) were collected in this study. Only bug reports with their resolution marked as fixed and the status marked as resolved, verified or closed were extracted. This extraction scheme will ensure the presence of developers who had actually fixed the bug. In this study, four most important bug meta-fields were used: component, severity, priority and operating system. These parameters are selected as they contain the most important information related to a bug and are extensively used in literature [13, 77]. Moreover, these fields generally do not contain any missing values for both fixed and new bug reports. This allows enough training and testing tokens for optimized bug report assignment. Initially, a total of 68,904 bug reports was obtained for all the four projects (20,483 bug reports for the Mozilla project, 39,758 for the Eclipse project, 6,326 for the Gnome project and 2,337 for the Open Office project). The collected bug reports were fixed by 3,301 unique developers (1,218 developers for the Mozilla project, 1,342 for the Eclipse project, 611 for the Gnome project and 130 for the Open Office project).



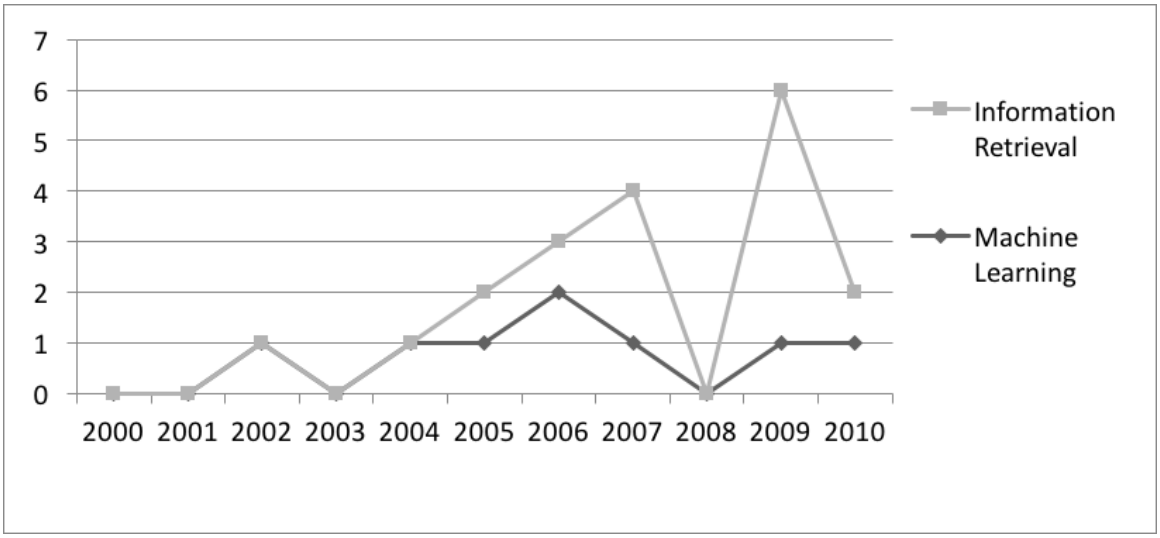
(a) Past studies



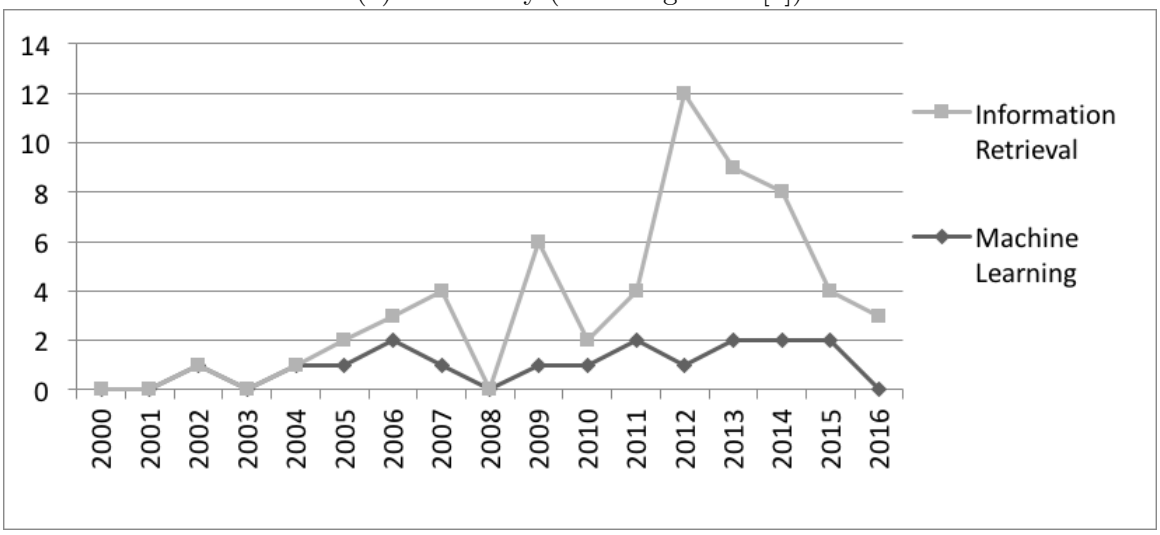
(b) Present study

Figure 5. Frequency distribution

For pre-processing, the bug reports in which the assigned-to field was unspecified were removed. In the Bugzilla repository, there are developers who had fixed few bugs. The inclusion of such developers would deteriorate the model performance so the parameter was further tuned, ($N \geq 10$), i.e., the number of bug reports fixed by a developer in the past. Hence, finally a total of 59,448 bug reports were obtained for all four projects (15,017 bug reports for the Mozilla project, 37,425 for the Eclipse project, 4,947 for the Gnome project and 2,059 for the Open Office project). The pre-processed bug reports were fixed by 940 unique developers (267 developers for the Mozilla project, 505 for the Eclipse project, 140 for the Gnome project and 28 for the Open Office project). Table 9 shows various details of the datasets used for



(a) Past study (T. Zhang et al. [2])



(b) Present study

Figure 6. Year wise distribution

comparison: start date, end date, number of collected bugs, number of distinct assignees (or developers), number of bug reports with $N \geq 10$, number of assignees who have fixed more than 10 bug reports in the past, unique number of tokens in various meta-fields of bug reports, such as component, severity, priority and operating system. A total 500 fixed bug reports randomly selected from each project were used for testing.

For the machine learning based classification, the use of four machine learning algorithms was investigated: Naïve Bayes, J48, Random tree and Bayes Net. These algorithms were selected as they covered different categories of supervised machine learning algorithms. The Weka toolkit was used for experimentation. Table 10 shows the results of the 10-fold cross validation of the machine learning based approach. Different classifiers achieved the best classification accuracy among different projects. For instance, in the Mozilla project the J48 classifier obtained the best classification accuracy of 44%, whereas the Naïve Bayes, Random Tree and Bayes net classifiers achieved 35%, 40% and 39% accuracy, respectively. Similarly, for the Eclipse project J48 and Random Tree obtained the best classification results of 44% accuracy. For the Gnome

Table 9. Dataset Details

	Mozilla	Eclipse	Gnome	Open Office
Start Date	01/01/2011	01/01/2011	01/01/2011	01/01/2011
End date	31/12/2016	31/12/2016	31/12/2016	31/12/2016
#bug reports collected	20,483	39,758	6,326	2,337
#assignees	1,218	1,342	611	130
#bug reports (N>=10)	15,017	37,425	4,947	2,059
#assignees(N>=10)	267	505	140	28
#component	367	498	400	100
#severity	7	7	7	6
#priority	5	5	5	5
#operating system	27	30	11	28

project, the J48 classifier obtained an accuracy of 53% and for the Open Office project the Random Tree classifier achieved the best accuracy of 45.2%. Overall, it was found out that a single classifier could not be declared as the best one for all the projects and different classifiers perform variably for different projects. However, it was observed that tree based classifiers, J48 and Random Tree are best suitable for bug report assignment.

For the information retrieval based technique, the term frequency (TF) based approach was used as it is most widely used in the literature [5,6,40]. First a term-author-matrix was created, $M[i,j]$, from the tokens obtained from the different meta-fields of bug reports (component, severity, priority and operating system). In the term-author-matrix, M denotes all the unique developers, i are authors and all the values in the various tokens in the meta-fields of a bug report are considered as terms, j. Each entry in the matrix represents the frequency, f_{ij} of developer, i with respect to a term, j. Frequency f_{ij} represents the expertise of a developer, i with respect to a term, j based on the work done by the developer in the past. Figure 7 shows an instance of a term-author-matrix. In the figure, gui, general, regression represents various distinct terms (or tokens) obtained from the various meta-fields of bug report and pollman, jaze and rick are the developers in the bug repository. The numeric values in the matrix represent the expertise values of developers while w.r.t. the terms in the past fixed bug reports.

To identify a suitable developer for a new bug report, its terms are extracted from the meta-fields and are considered as a search query. Columns from term-author-matrix matching the terms in the search query are extracted. To calculate the final expertise score for each developer, the frequency values of each developer are aggregated. The developer with a higher score is considered to be suitable as they have more expertise in the areas of the new bug report. Table 10 shows the results of the top-k ($k = 5$ and 10) recommendation list sizes in the informational retrieval based approach. In the Mozilla project, the achieved maximum accuracy is 52% for the top-10 list size. Similarly, the maximum achieved accuracy is 49.6%, 72% and 87% for the Eclipse, Gnome and Open Office projects, respectively.

Comparing the results of the machine learning and information retrieval based techniques, it was found out that the information retrieval based techniques yield better accuracy as compared to the machine learning based technique. Thus, information retrieval is a better technique for activity profile based bug report assignment approaches. In the Mozilla project, the J48 machine learning algorithm gives 44% accuracy which increases by 6% for the top-10 recommendation list in information retrieval. Similarly, in the Eclipse, Gnome and Open Office projects the accuracy of the information retrieval based technique is significantly higher than in the machine learning based approach. This supports the view that the information retrieval based technique achieves better accuracy and thus there is a trend shift in bug assignment approaches from the machine learning based techniques to the information retrieval based technique.

Developers/ Authors	Terms: gui	general	regression
Pollman	8	6	5
Jaze	4	1	8
Rick	3	4	2

Expert values of developers w.r.t. terms

Figure 7. An instance of term-author-matrix

Table 10. Classification accuracy of Machine Learning and Information Retrieval algorithms

		Mozilla	Eclipse	Gnome	Open Office
Machine Learning	Naïve Bayes	35%	33%	43%	44.2%
	J48	44%	44%	53%	42.3%
	Random Tree	40%	44%	52%	45.2%
	Bayes Net	39%	35%	47%	44.2%
Information Retrieval	Top-5	47%	45.2%	60%	62%
	Top-10	52%	49.6%	72%	87%

5. Conclusion and future work

Bug report assignment is a time consuming and tedious task for a bug triager. This paper presents a review and classification of 75 research papers in the area of automated bug assignment. Seven categories of bug assignment approaches have been identified in this study. The identified categories are machine learning, information retrieval, auction, social network, tossing graph, fuzzy set and operational research based techniques. We systematically organized 75 surveyed papers in one of the seven identified techniques of bug triaging. Further, we analysed the surveyed papers in two perspectives: the frequency wise distribution of techniques and the year wise distribution of each technique. Interesting facts are captured in this analytical study. First, the machine learning and information retrieval based techniques are most popular for automatic bug report assignment. Second, the current trend of bug assignment approaches is shifting from machine learning to the information retrieval based techniques.

To examine the reason behind this shift, an empirical study was performed on the machine learning and information retrieval based bug triaging technique. The study was done on real time, large scale, open source projects, Mozilla, Eclipse, Gnome and Open Office. The results of the analysis showed an increase of up to 12.8% in the efficiency for the top-5 list size in the information retrieval based technique. Thus, the information retrieval based techniques are the best choice for bug triaging. The possible reasons for this shift are better efficiency, ability to consider the current expertise of developers and the ability to cooperate with other techniques.

Although a high volume of literature is available in the area of automated bug assignment, there is still a deficiency of a technique which presents an acceptable efficiency to be used in the real time environment. There are three major difficulties in bug handling. a) Sheer volume of information available in bug repositories, b) collaborative work by developers for bug rectification, and c) continuous evolvement of project or software systems. These difficulties lead to a series of open issues in bug assignment approaches, such as profiling new developers, maintaining updated profiles, workload balancing, assignment of reopened bugs and most importantly the reliability of the data in bug tracking repositories. In the future, we plan to implement an information retrieval based technique which considers the time-based expertise computation and to test it on a large dataset considering a huge number of developers as is the case in real time environment.

References

[1] J. Zhang, X. Wang, D. Hao, B. Xie, L. Zhang, and H. Mei, "A survey on bug-report analysis," *Science China Information Sciences*, Vol. 58, No. 2, 2015, pp. 1–24.

[2] T. Zhang, H. Jiang, X. Luo, and A.T. Chan, "A literature review of research in bug resolution: Tasks, challenges and future directions," *The Computer Journal*, Vol. 59, No. 5, 2016, pp. 741–773.

[3] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Software Engineering Group, School of Computer Science and Mathematics, Keele University and Department of Computer Science, University of Durham, Tech. Rep. EBSE 2007-001, 2007. [Online]. https://pdfs.semanticscholar.org/e62d/bbbbe70cabcde3335765009e94ed2b9883d5.pdf

[4] D. Matter, A. Kuhn, and O. Nierstrasz, "Assigning bug reports using a vocabulary-based expertise model of developers," in *6th IEEE International Working Conference on Mining Software Repositories*. IEEE, 2009, pp. 131–140.

[5] A.S.K. Singh, "Bug triaging: Profile oriented developer recommendation," *International Journal of Innovative Research in Advanced Engineering*, Vol. 2, 2014, pp. 36–42.

[6] H. Naguib, N. Narayan, B. Brügge, and D. Helal, "Bug report assignee recommendation using activity profiles," in *10th IEEE Working Conference on Mining Software Repositories (MSR)*. IEEE, 2013, pp. 22–30.

[7] I. Aljarah, S. Banitaan, S. Abufardeh, W. Jin, and S. Salem, "Selecting discriminating terms for bug assignment: a formal analysis," in *Proceedings of the 7th International Conference on Predictive Models in Software Engineering*. ACM, 2011.

[8] A. Sureka, H. Kumar Singh, M. Bagewadi, A. Mitra, and R. Karanth, "A decision support platform for guiding a bug triager for resolver recommendation using textual and non-textual features," in *3rd International Workshop on Quantitative Approaches to Software Quality*, 2015, p. 25.

[9] R. Shokripour, J. Anvik, Z.M. Kasirun, and S. Zamani, "Why so complicated? simple term filtering and weighting for location-based bug report assignment recommendation," in *Proceedings of the 10th Working Conference on Mining Software Repositories*. IEEE Press, 2013, pp. 2–11.

[10] F. Servant and J.A. Jones, "WhoseFault: automatic developer-to-fault assignment through fault localization," in *Proceedings of the 34th International Conference on Software Engineering*. IEEE Press, 2012, pp. 36–46.

[11] G. Murphy and D. Cubranic, "Automatic bug triage using text categorization," in *Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering*, 2004.

[12] J. Xuan, H. Jiang, Z. Ren, J. Yan, and Z. Luo, "Automatic bug triage using semi-supervised text classification," in *22nd International Conference on Software Engineering and Knowledge Engineering (SEKE)*, 2010, pp. 209–214.

[13] J. Anvik, "Automating bug report assignment," in *Proceedings of the 28th International Conference on Software Engineering*. ACM, 2006, pp. 937–940.

[14] J. Anvik, L. Hiew, and G.C. Murphy, "Coping with an open bug repository," in *Proceedings of the 2005 OOPSLA Workshop on Eclipse Technology Exchange*. ACM, 2005, pp. 35–39.

[15] J. Anvik and G.C. Murphy, "Reducing the effort of bug report triage: Recommenders for development-oriented decisions," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, Vol. 20, No. 3, 2011, pp. 10:1–10:35.

[16] J.K. Anvik, "Assisting bug report triage through recommendation," Ph.D. dissertation, University of British Columbia, 2007.

[17] G.A. Di Lucca, M. Di Penta, and S. Gradara, "An approach to classify software maintenance requests," in *International Conference on Software Maintenance*. IEEE, 2002, pp. 93–102.

[18] P. Bhattacharya, I. Neamtiu, and C.R. Shelton, "Automated, highly-accurate, bug assignment using machine learning and tossing graphs," *Journal of Systems and Software*, Vol. 85, No. 10, 2012, pp. 2275–2292.

[19] H. Hu, H. Zhang, J. Xuan, and W. Sun, "Effective bug triage based on historical bug-fix information," in *IEEE 25th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2014, pp. 122–132.

[20] J. Xuan, H. Jiang, Y. Hu, Z. Ren, W. Zou, Z. Luo, and X. Wu, "Towards effective bug triage with software data reduction techniques," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 27, No. 1, 2015, pp. 264–280.

[21] X. Xia, D. Lo, X. Wang, and B. Zhou, "Accurate developer recommendation for bug resolution," in *20th Working Conference on Reverse Engineering (WCRE)*. IEEE, 2013, pp. 72–81.

[22] M.R. Karim, G. Ruhe, M. Rahman, V. Garousi, T. Zimmermann *et al.*, "An empirical investigation of single-objective and multiobjective evolutionary algorithms for developer's assignment to bugs," *Journal of Software: Evolution and Process*, Vol. 28, No. 12, 2016, pp. 1025–1060.

[23] A. Moin and G. Neumann, "Assisting bug triage in large open source projects using approximate string matching," in *Seventh International Conference on Software Engineering Advances (ICSEA)*, Lisbon, Portugal, 2012.

[24] T. Zhang and B. Lee, "A hybrid bug triage algorithm for developer recommendation," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. ACM, 2013, pp. 1088–1094.

[25] S.N. Ahsan, J. Ferzund, and F. Wotawa, "Automatic software bug triage system (BTS) based on latent semantic indexing and support vector machine," in *Fourth International Conference on Software Engineering Advances*. IEEE, 2009, pp. 216–221.

[26] G. Canfora and L. Cerulo, "How software repositories can help in resolving a new change request," in *IEEE International Workshop on Software Technology and Engineering Practice (STEP)*, 2005, pp. 99–103.

[27] H. Kagdi, M. Gethers, D. Poshyvanyk, and M. Hammad, "Assigning change requests to software developers," *Journal of Software: Evolution and Process*, Vol. 24, No. 1, 2012, pp. 3–33.

[28] N.K. Nagwani and S. Verma, "Predicting expert developers for newly reported bugs using frequent terms similarities of bug attributes," in *9th International Conference on ICT and Knowledge Engineering (ICT & Knowledge Engineering)*. IEEE, 2012, pp. 113–117.

[29] O. Baysal, M.W. Godfrey, and R. Cohen, "A bug you like: A framework for automated assignment of bugs," in *IEEE 17th International Conference on Program Comprehension*. IEEE, 2009, pp. 297–298.

[30] K. Kevic, S.C. Müller, T. Fritz, and H.C. Gall, "Collaborative bug triaging using textual similarities and change set analysis," in *6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, 2013, pp. 17–24.

[31] X. Xie, W. Zhang, Y. Yang, and Q. Wang, "Dretom: Developer recommendation based on topic models for bug resolution," in *Proceedings of the 8th International Conference on Predictive Models in Software Engineering*. ACM, 2012, pp. 19–28.

[32] M. Alenezi, K. Magel, and S. Banitaan, "Efficient bug triaging using text mining," *JSW*, Vol. 8, No. 9, 2013, pp. 2185–2190.

[33] G. Canfora and L. Cerulo, "Supporting change request assignment in open source development," in *Proceedings of the 2006 ACM Symposium on Applied Computing*. ACM, 2006, pp. 1767–1772.

[34] T. Zhang, G. Yang, B. Lee, and E.K. Lua, "A novel developer ranking algorithm for automatic bug triage using topic model and developer relations," in *21st Asia-Pacific Software Engineering Conference (APSEC)*, Vol. 1. IEEE, 2014, pp. 223–230.

[35] E. Linstead, P. Rigor, S. Bajracharya, C. Lopes, and P. Baldi, "Mining eclipse developer contributions via author-topic models," in *Fourth International Workshop on Mining Software Repositories*. IEEE, 2007, pp. 30–30.

[36] G. Yang, T. Zhang, and B. Lee, "Towards semi-automatic bug triage and severity prediction based on topic model and multi-feature of bug reports," in *IEEE 38th Annual Computer Software and Applications Conference (COMPSAC)*. IEEE, 2014, pp. 97–106.

[37] X. Xia, D. Lo, Y. Ding, J.M. Al-Kofahi, T.N. Nguyen, and X. Wang, "Improving automated bug triaging with specialized topic model," *IEEE Transactions on Software Engineering*, Vol. 43, No. 3, 2017, pp. 272–297.

[38] K. Somasundaram and G.C. Murphy, "Automatic categorization of bug reports using latent dirichlet allocation," in *Proceedings of the 5th India Software Engineering Conference*. ACM, 2012, pp. 125–130.

[39] R. Shokripour, Z.M. Kasirun, S. Zamani, and J. Anvik, "Automatic bug assignment using information extraction methods," in *International Conference on Advanced Computer Science Applications and Technologies (ACSAT)*. IEEE, 2012, pp. 144–149.

[40] R. Shokripour, J. Anvik, Z.M. Kasirun, and S. Zamani, "A time-based approach to automatic bug report assignment," *Journal of Systems and Software*, Vol. 102, 2015, pp. 109–122.

[41] D. Mohan, N. Sardana *et al.*, "Visheshagya: Time based expertise model for bug report assignment," in *Ninth International Conference on Contemporary Computing (IC3)*. IEEE, 2016, pp. 1–6.

[42] S. Zamani, S.P. Lee, R. Shokripour, and J. Anvik, "A noun-based approach to feature location using time-aware term-weighting," *Information*

and Software Technology, Vol. 56, No. 8, 2014, pp. 991–1011.

[43] T.T. Nguyen, A.T. Nguyen, and T.N. Nguyen, "Topic-based, time-aware bug assignment," *ACM SIGSOFT Software Engineering Notes*, Vol. 39, No. 1, 2014, pp. 1–4.

[44] H. Hosseini, R. Nguyen, and M.W. Godfrey, "A market-based bug allocation mechanism using predictive bug lifetimes," in *16th European Conference on Software Maintenance and Reengineering (CSMR)*. IEEE, 2012, pp. 149–158.

[45] W. Wu, W. Zhang, Y. Yang, and Q. Wang, "Drex: Developer recommendation with k-Nearest-Neighbor search and expertise ranking," in *18th Asia Pacific Software Engineering Conference (APSEC)*. IEEE, 2011, pp. 389–396.

[46] T. Zhang and B. Lee, "An automated bug triage approach: A concept profile and social network based developer recommendation," in *International Conference on Intelligent Computing*. Springer, 2012, pp. 505–512.

[47] J. Xuan, H. Jiang, Z. Ren, and W. Zou, "Developer prioritization in bug repositories," in *34th International Conference on Software Engineering (ICSE)*. IEEE, 2012, pp. 25–35.

[48] G. Yang, T. Zhang, and B. Lee, "Utilizing a multi-developer network-based developer recommendation algorithm to fix bugs effectively," in *Proceedings of the 29th Annual ACM Symposium on Applied Computing*. ACM, 2014, pp. 1134–1139.

[49] W. Zhang, S. Wang, Y. Yang, and Q. Wang, "Heterogeneous network analysis of developer contribution in bug repositories," in *International Conference on Cloud and Service Computing (CSC)*. IEEE, 2013, pp. 98–105.

[50] P. Bhattacharya and I. Neamtiu, "Fine-grained incremental learning and multi-feature tossing graphs to improve bug triaging," in *IEEE International Conference on Software Maintenance (ICSM)*. IEEE, 2010, pp. 1–10.

[51] G. Jeong, S. Kim, and T. Zimmermann, "Improving bug triage with bug tossing graphs," in *Proceedings of the the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*. ACM, 2009, pp. 111–120.

[52] L. Chen, X. Wang, and C. Liu, "An approach to improving bug assignment with bug tossing graphs and bug similarities," *JSW*, Vol. 6, No. 3, 2011, pp. 421–427.

[53] A. Tamrawi, T.T. Nguyen, J. Al-Kofahi, and T.N. Nguyen, "Fuzzy set-based automatic bug triaging: NIER track," in *33rd International Conference on Software Engineering (ICSE)*. IEEE, 2011, pp. 884–887.

[54] A. Tamrawi, T.T. Nguyen, J.M. Al-Kofahi, and T.N. Nguyen, "Fuzzy set and cache-based approach for bug triaging," in *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*. ACM, 2011, pp. 365–375.

[55] A. Niknafs, J. Denzinger, and G. Ruhe, "A systematic literature review of the personnel assignment problem," in *Proceedings of the International Multiconference of Engineers and Computer Scientists*, 2013.

[56] M.M. Rahman, S. Sohan, F. Maurer, and G. Ruhe, "Evaluation of optimized staffing for feature development and bug fixing," in *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*. ACM, 2010, p. 42.

[57] D. Panagiotou, F. Paraskevopoulos, and L. Stojanovic, Specifications of developer profile, (2010). [Online]. http://www.alert-project.eu/sites/portal2-alert.atosorigin.es/files/content-files/download/Specification%20of%20Developer%20Profile.pdf

[58] N.K. Nagwani and S. Verma, "Rank-me: A Java tool for ranking team members in software bug repositories," *Journal of Software Engineering and Applications*, Vol. 5, 2012, pp. 255–261.

[59] D. Kim, Y. Tao, S. Kim, and A. Zeller, "Where should we fix this bug? a two-phase recommendation model," *IEEE Transactions on Software Engineering*, Vol. 39, No. 11, 2013, pp. 1597–1610.

[60] M. Linares-Vásquez, K. Hossen, H. Dang, H. Kagdi, M. Gethers, and D. Poshyvanyk, "Triaging incoming change requests: Bug or commit history, or code authorship?" in *28th IEEE International Conference on Software Maintenance (ICSM)*. IEEE, 2012, pp. 451–460.

[61] B. Ashok, J. Joy, H. Liang, S.K. Rajamani, G. Srinivasa, and V. Vangala, "DebugAdvisor: a recommender system for debugging," in *Proceedings of the the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*. ACM, 2009, pp. 373–382.

[62] H. Kagdi and D. Poshyvanyk, "Who can help me with this change request?" in *17th International Conference on Program Comprehension*. IEEE, 2009, pp. 273–277.

[63] J. Anvik and G.C. Murphy, "Determining implementation expertise from bug reports," in *Proceedings of the Fourth International Workshop on Mining Software Repositories*. IEEE Computer Society, 2007, p. 2.

[64] S. Minto and G.C. Murphy, "Recommending emergent teams," in *Fourth International Workshop on Mining Software Repositories*. IEEE, 2007, pp. 5–12.

[65] G. Gousios, E. Kalliamvakou, and D. Spinellis, "Measuring developer contribution from software repository data," in *Proceedings of the 2008 International Working Conference on Mining Software Repositories*. ACM, 2008, pp. 129–132.

[66] T. Zhang, G. Yang, B. Lee, and A.T. Chan, "Guiding bug triage through developer analysis in bug reports," *International Journal of Software Engineering and Knowledge Engineering*, Vol. 26, No. 03, 2016, pp. 405–431.

[67] T. Zhang, G. Yang, B. Lee, and I. Shin, "Role analysis-based automatic bug triage algorithm," IPSJ SIG Technical Report, Tech. Rep., 2012.

[68] X. Xia, D. Lo, X. Wang, and B. Zhou, "Dual analysis for recommending developers to resolve bugs," *Journal of Software: Evolution and Process*, Vol. 27, No. 3, 2015, pp. 195–220.

[69] J. Helming, H. Arndt, Z. Hodaie, M. Koegel, and N. Narayan, "Automatic assignment of work items," in *International Conference on Evaluation of Novel Approaches to Software Engineering*. Springer, 2010, pp. 236–250.

[70] W. Zou, Y. Hu, J. Xuan, and H. Jiang, "Towards training set reduction for bug triage," in *IEEE 35th Annual Computer Software and Applications Conference (COMPSAC)*. IEEE, 2011, pp. 576–581.

[71] Z. Lin, F. Shu, Y. Yang, C. Hu, and Q. Wang, "An empirical study on bugassignment automation using Chinese bug data," in *3rd International Symposium on Empirical Software Engineering and Measurement*. IEEE, 2009, pp. 451–455.

[72] S. Banitaan and M. Alenezi, "Tram: An approach for assigning bug reports using their metadata," in *Third International Conference on Communications and Information Technology*. IEEE, 2013, pp. 215–219.

[73] J. Anvik, L. Hiew, and G.C. Murphy, "Who should fix this bug?" in *Proceedings of the 28th International Conference on Software Engineering*. ACM, 2006, pp. 361–370.

[74] M. Sharma, M. Kumari, and V. Singh, "Bug assignee prediction using association rule mining," in *International Conference on Computational Science and Its Applications*. Springer, 2015, pp. 444–457.

[75] J. Park, M. Lee, J. Kim, S. Hwang, and S. Kim, "Costriage: A cost-aware triage algorithm for bug reporting systems," in *Proceedings of the National Conference on Artificial Intelligence*, 2011, p. 139.

[76] M.M. Rahman, G. Ruhe, and T. Zimmermann, "Optimized assignment of developers for fixing bugs an initial evaluation for eclipse projects," in *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*. IEEE Computer Society, 2009, pp. 439–442.

[77] R.K. Saha, S. Khurshid, and D.E. Perry, "An empirical study of long lived bugs," in *Software Evolution Week–IEEE Conference on Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE)*. IEEE, 2014, pp. 144–153.