International Journal of Advanced Research in Education and TechnologY(IJARETY)

| ISSN: 2394-2975 | www.ijarety.in| A Bi-Monthly, Double-Blind Peer Reviewed & Referred Journal |

|| Volume 9, Issue 2, March 2022 ||

Effective Management of Bugs in Software Repositories

Kaviya V, Reshma K Rajan

Department of Computer Engineering, MBC Engineering College, Kuttikanam, Kerala, India

ABSTRACT: Software organizations spend most of cost in managing programming bugs. An unavoidable stride of settling bugs is bug triage, which plans to accurately dole out an engineer to another bug. To diminish the time cost in manual work, content order strategies are connected to direct programmed bug triage. In this paper, we address the problem of information lessening for bug triage, i.e., how to decrease the scale and enhance the nature of bug data. We consolidate case choice with highlight choice to all the while decrease information scale on the bug measurement and the word measurement. To decide the request of applying occurrence determination and highlight choice, we remove qualities from recorded bug informational collections and fabricate a prescient model for another bug informational index. Our work gives a way to deal with utilizing procedures on information preparing to shape decreased and superb bug information in programming advancement and upkeep.

KEYWORDS: Mining programming vaults, utilization of information preprocessing, information administration in bug stores, bug information lessening, include determination, case choice, bug triage, forecast for decrease orders .

I. Introduction

Mining programming storehouses is an interdisciplinary space, which plans to utilize information mining to bargain with programming designing issues. In cutting edge programming advancement, programming storehouses are expansive scale databases for putting away the yield of programming advancement, e.g., source code, bugs, messages, and details. Conventional programming examination is not totally appropriate for the extensive scale and complex information in programming storehouses. Information mining has risen as a promising intends to deal with programming information.

By utilizing information mining methods, mining programming vaults can reveal intriguing data in programming stores and settle real world programming issues. A bug storehouse (a commonplace programming archive, for putting away points of interest of bugs), assumes a vital part in overseeing programming bugs. Programming bugs are unavoidable and settling bugs is costly in programming improvement. Programming organizations spend more than 45 percent of cost in settling bugs. Expansive programming ventures send bug vaults (additionally called bug following frameworks) to bolster data accumulation and to help engineers to deal with bugs [2].

In a bug store, a bug is kept up as a bug report, which records the literary depiction of recreating the bug and redesigns agreeing to the status of bug settling [27]. A bug vault gives a information stage to bolster many sorts of undertakings on bugs, e.g., blame forecast bug confinement and reopened bug investigation [26]. In this paper, bug reports in a bug store are called bug information. There are two difficulties identified with bug information that may influence the successful utilization of bug storehouses in programming advancement assignments, to be specific the huge scale and the low quality. On one hand, because of the day by day detailed bugs, an expansive number of new bugs are put away in bug storehouses. A set of users can upload files after registering into the site. Developers who are inactive will be deactivated by the classifier. Developers resolves the bug reports by specifying the solution in the specified area. New bugs will be added into the bug data set by the classifier [10].

Two average attributes of low-quality bugs are commotion and excess. Loud bugs may misdirect related engineers [27] while excess bugs squander the restricted time of bug taking care of [22].

A tedious stride of taking care of programming bugs is bug triage, which plans to relegate a right designer to settle another bug [1],[7],[13]. In customary programming improvement, new bugs are physically triaged by a specialist designer, i.e., a human triager. Because of the vast number of every day bugs and the absence of aptitude of the considerable number of bugs, manual bug triage is costly in time cost and low in precision. To maintain a strategic distance from the costly cost of manual bug triage, existing work [1] has proposed a programmed bug triage approach, which applies content grouping systems to foresee designers for bug reports. In this approach, a bug report is mapped to an archive and a related designer is mapped to the mark of the archive. At that point, bug triage is changed over into an issue of content arrangement what's more, is consequently illuminated with develop content arrangement strategies.

In view of the aftereffects of content characterization, a human triager doles out new bugs by consolidating his/her skill. To enhance the precision of content characterization strategies for bug triage, some further strategies are researched, Nonetheless, substantial scale and low-quality bug information in bug vaults hinder the strategies of programmed bug triage. Since programming bug information are a sort of freestyle content information (produced by engineers), it is important to create all around prepared bug information to encourage the application [29]. In this paper, we address the issue of information decrease for bug triage, i.e., how to decrease the bug information to spare the work cost of designers and enhance the quality to encourage the procedure of bug triage. Information decrease for bug triage intends to fabricate a little scale and high caliber set of bug information by expelling bug reports and words, which are excess or non-instructive.

In our work, we consolidate existing strategies of occasion choice and highlight choice to at the same time diminish the bug measurement what's more, the word measurement. The decreased bug information contain less bug reports and less words than the first bug information and give comparable data over the unique bug information. We assess the diminished bug information as indicated by two criteria: the size of an informational index and the exactness of bug triage. To stay away from the predisposition of a solitary calculation, we experimentally look at the aftereffects of four example determination calculations and four element choice calculations.

Given a case choice calculation and an element determination calculation, the request of applying these two calculations may influence the consequences of bug triage. In this paper, we propose a prescient model to decide the request of applying example choice and highlight determination. We allude to such assurance as expectation for decrease orders. Drawn on the encounters in programming metrics,1 we separate the properties from chronicled bug informational collections. At that point, we prepare a parallel classifier on bug informational collections with removed traits and anticipate the request of applying example determination and highlight choice for another bug informational collection.

Test comes about demonstrate that applying the case choice method to the informational index can diminish bug reports yet the precision of bug triage might be diminished; applying the component choice method can diminish words in the bug information and the precision can be expanded. In light of the traits from authentic bug informational indexes, our prescient model can give the exactness of 71.8 percent for foreseeing the lessening arrange.

In view of top hub investigation of the qualities, comes about demonstrate that no individual quality can decide the diminishment arrange and every ascribe is useful to the forecast. The essential commitments of this paper are as per the following:

1) We show the issue of information diminishment for bug triage. This issue expects to increase the informational collection of bug triage in two viewpoints, in particular a) to all the while lessen the sizes of the bug measurement and the word measurement and b) to enhance the exactness of bug triage.

2) We propose a blend way to deal with tending to the issue of information lessening. This can be seen as an utilization of occurrence determination and highlight choice in bug vaults.

3) We assemble a double classifier to foresee the request of applying occurrence determination and highlight choice.

To our insight, the request of applying case choice what's more, element determination has not been researched in related areas. This paper is an augmentation of our past work [25]. In this augmentation, we include new traits separated from bug informational collections, expectation for lessening requests, and examinations on four occasion determination calculations, four element choice calculations, and their blends.

II. BACKGROUND AND MOTIVATION

A. *BACKGROUND*

Bug stores are broadly utilized for looking after programming bugs, e.g., a well known and open source bug storehouse, Bugzilla. Once a product bug is found, a columnist (commonly a designer, an analyzer, or an end client) records this bug to the bug store. In a bug report, the synopsis and the depiction are two key things about the data of the bug, which are recorded in regular dialects. As their names propose, the outline indicates a general proclamation for distinguishing a bug while the portrayal gives the points of interest for recreating the bug.

Some different things are recorded in a bug report for encouraging the distinguishing proof of the bug, such as the item, the stage, and the significance. Once a bug report is framed, a human triager doles out this bug to a designer, who will attempt to settle this bug. This engineer is recorded in a thing allocated to. The doled out to will change to another engineer if the already appointed engineer can't settle this bug. The way toward appointing a adjust designer for settling the bug is called bug triage.

An engineer, who is doled out to another bug report, begins to settle the bug in view of the information of authentic bug settling [12], [27]. Regularly, the designer pays endeavours to get it the new bug report and to look at verifiably settled bugs as a kind of perspective (e.g., hunting down comparable bugs [54] and applying existing answers for the new bug [10]). A thing status of a bug report is changed by the present consequence of taking care of this bug until the bug is totally settled. Changes of a bug report are put away in any thing history. This bug has been allotted to three designers and as it were the last designer can deal with this bug accurately.

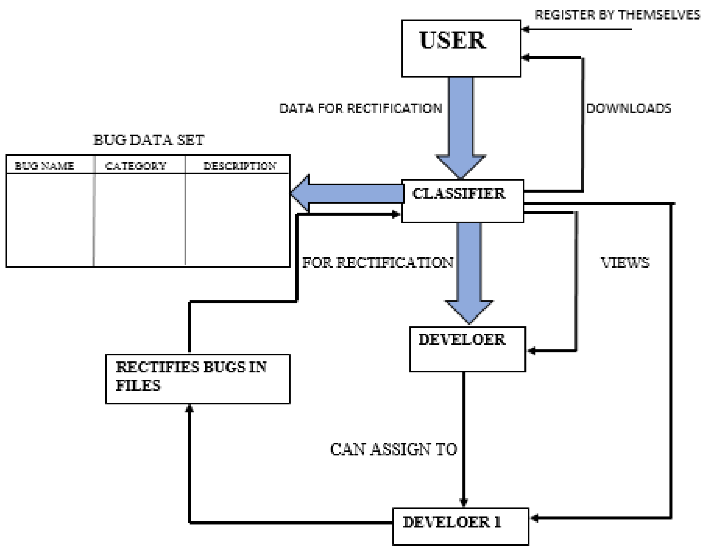

Fig 2.1:Reduction of bug data for bug triage

Evolving designers goes on for more than seven months while settling this bug just costs three days. Manual bug triage by a human triager is time consuming what's more, mistake inclined since the quantity of day by day bugs is vast to effectively allot and a human triager is difficult to ace the learning about every one of the bugs [12]. Existing work utilizes the methodologies in view of content characterization to help bug triage, e.g., [1], [7]. In such methodologies, the synopsis and the portrayal of a bug report are removed as the printed content while the engineer who can settle this bug is set apart as the mark for arrangement.

According to Fig 2.1 classifier creates the bug data set which contains attributes such as bug name, category and bug description. A developer can assign a bug report to another developer if he could not able to solve it. Developers solves the bug report assigned to him and returns back to the classifier. In points of interest, existing bug reports with their engineers are shaped as a preparation set to prepare a classifier, new bug reports are dealt with as a test set to look at the aftereffects of the characterization.**.** we outline the essential system of bug triage in light of content order. As appeared in Fig 2.1, we see a bug informational index as a content framework. Every line of the grid demonstrates one bug report while every segment of the grid demonstrates single word. To stay away from the low precision of bug triage, a suggestion list with the size k is utilized to give a rundown of k engineers, who have the top-k probability to settle the new bug.

III. MOTIVATION

Genuine information dependably incorporate clamour and repetition. Loud information may deceive the information investigation methods [29] while excess information may build the cost of information preparing [4]. In bug stores, all the bug reports are filled by designers in regular dialects. The low-quality bugs aggregate in bug stores with the development in scale. Such huge scale and low-quality bug information may break down the adequacy of settling bugs [10], [27].

To concentrate the boisterous bug report, we take the bug report of bug 201598 as Example 2 (Note that both the outline and the portrayal are incorporated). Case 2 (Bug 201598). 3.3.1 about says 3.3.0. Manufacture id: M20070829-0800. 3.3.1 about says 3.3.0. This bug report exhibits the mistake in the adaptation discourse. Be that as it may, the points of interest are not clear. Unless an engineer is exceptionally acquainted with the foundation of this bug, it is elusive the points of interest. As indicated by the thing history, this bug is settled by the engineer who has revealed this bug. Be that as it may, the rundown of this bug may make different designers confounded.

Besides, from the point of view of information handling, particularly programmed preparing, the words in this bug might be evacuated since these words are not useful to recognize this bug. Along these lines, it is important to evacuate the uproarious bug reports what's more, words for bug triage. To concentrate the repetition between bug reports, we list two bug reports of bugs 200019 and 204653 in Example 3 (the things portrayal are precluded). Case 3. Bugs 200019 and 204653. (Bug 200019) Argument popup not highlighting the amend contention . . . (Bug 204653)

Argument highlighting wrong . . . In bug stores, the bug report of bug 200019 is set apart as a copy one of bug 204653 (a copy bug report, indicates that a bug report depicts one programming blame, which has a similar underlying driver as a current bug report [22]).

The printed substance of these two bug reports are comparable. Henceforth, one of these two bug reports might be picked as the agent one. In this way, we need to utilize a certain method to expel one of these bug reports. Along these lines, a procedure to expel additional bug reports for bug triage is required. In light of the over three illustrations, it is important to propose a way to deal with diminishing the scale and increasing the nature of bug information

IV. DATA REDUCTION FOR BUG TRIAGE

Persuaded by the three cases in Section 2.2, we propose bug information decrease to lessen the scale and to enhance the nature of information in bug stores. Fig 2.1 outlines the bug information diminishment in our work, which is connected as a stage in information arrangement of bug triage. We consolidate existing procedures of example determination also, include determination to evacuate certain bug reports and words. An issue for lessening the bug information is to decide the request of applying example choice and include choice, which is indicated as the expectation of lessening orders**.** In this area, we first present how to apply occasion determination and highlight choice to bug information, i.e., information lessening for bug triage. At that point, we list the advantage of the information lessening. The points of interest of the forecast for diminishment orders will be appeared in Section 4.

```
Algorithm 1:
Information decrease in view of FS -> IS
Input: preparing set T with n words and m bug reports,
decrease arrange FS -> IS last number $n_F$ of words,
last number $m_I$ of bug reports,
Output: decreased informational index $T_{FI}$ for bug triage
1) apply FS to n expressions of T and figure target values for every one of the words;
2) select the top $n_F$ expressions of T and create a preparation set $T_F$ ;
3) apply IS to $m_I$ bug reports of $T_F$ ;
4) end IS the point at which the quantity of bug reports is equivalent to alternately not as much
as $m_I$ and produce the last preparing set $T_{FI}$ .
```

i. Applying Instance Selection and Feature Selection

In bug triage, a bug informational collection is changed over into a content lattice with two measurements, specifically the bug measurement and the word measurement. In our work, we use the blend of example determination and highlight choice to create a decreased bug informational collection. We supplant the first informational index with the decreased informational collection for bug triage. Occurrence determination and highlight choice are broadly utilized methods in information preparing. For a given informational index in a certain application, case choice is to get a subset of pertinent occasions (i.e., bug reports in bug information) while include choice plans to acquire a subset of significant elements (i.e., words in bug information) [4]. In our work, we utilize the blend of occasion choice and highlight choice. To recognize the requests of applying example choice and highlight choice, we give the accompanying indication. Given an occurrence choice calculation IS and an element determination calculation FS, we utilize FS -> IS to indicate the bug information decrease, which first applies FS and afterward IS; then again, IS -> FS indicates first applying IS and afterward FS. In Algorithm 1, we quickly show how to diminish the bug information in light of FS -> IS.

Given a bug informational index, the yield of bug information diminishment is another and lessened informational collection. Two calculations FS and IS are connected consecutively. Take note of that in Step 2), some of bug reports might be clear amid highlight i.e., every one of the words in a bug report are evacuated. Such clear bug reports are likewise expelled in the component choice. In our work, FS -> IS and IS -> FS are seen as two requests of bug information diminishment. To dodge the inclination from a solitary calculation, we inspect aftereffects of four common calculations of case choice and highlight determination, individually.

We quickly present these calculations as takes after. Example choice is a system to diminish the quantity of cases by expelling loud and repetitive occurrences . A case choice calculation can give a decreased informational collection by expelling non-agent cases [28]. As indicated by a current correlation think about and an existing survey , we pick four example determination calculations, to be specific Iterative Case Filter (ICF), Learning Vectors Quantization (LVQ) [9], Decremental Reduction Advancement Procedure (DROP) and Patterns by Requested

Projections (POP) [14]. Highlight determination is a preprocessing procedure for selecting a diminished arrangement of components for vast scale informational collections [4]. The diminished set is considered as the delegate highlights of the first list of capabilities. Since bug triage is changed over into content grouping, we concentrate on the component determination calculations in content information. In this paper, we pick four very much performed calculations in content information [16], and programming information to be specific Information Gain (IG) [6], x2 measurement (CH) Symmetrical Uncertainty property assessment (SU) [51], and Relief-F Attribute determination (RF) [15]. In view of highlight choice, words in bug reports are sorted concurring to their element values and a given number of words with expansive qualities are chosen as agent components.

ii. Benefit of Data Reduction

In our work, to spare the work cost of designers, the information diminishment for bug triage has two objectives,

1) lessening the information scale and

2) enhancing the precision of bug triage. Interestingly to demonstrating the literary substance of bug reports in existing work.

iii. Reducing the Data Scale

We lessen sizes of informational collections to spare the work cost of engineers. Bug measurement. As specified in Section 2.1, the point of bug triage is to appoint designers for bug settling. Once an engineer is doled out to another bug report, the designer can analyse verifiably settled bugs to frame an answer for the current bug report [12], [27]. For instance, recorded bugs are checked to recognize whether the new bug is the copy of a current one [22]; in addition, existing answers for bugs can be sought and connected to the new bug [10]. In this manner, we consider diminishing copy and boisterous bug reports to diminish the quantity of authentic bugs.

Practically speaking, the work cost of designers (i.e., the cost of looking at verifiable bugs) can be spared by diminishing the quantity of bugs based on occasion choice. Word measurement. We utilize highlight choice to evacuate uproarious on the other hand copy words in an informational collection. In light of highlight choice, the diminished informational collection can be taken care of all the more effortlessly by programmed procedures (e.g., bug triage approaches) than the unique informational index. Other than bug triage, the decreased informational index can be further utilized for other programming assignments after bug triage.

iv. Improving the Accuracy

Precision is a critical assessment foundation for bug triage. In our work, information decrease investigates and expels loud or copy data in informational indexes (see cases in Section 2.2). Bug measurement. Example determination can evacuate uninformative bug reports; then, we can watch that the exactness might be diminished by evacuating bug reports.

Table 1: Bug Data Set

Catgory	Bug Name	Description
User Interface Error	Missing/Wrong Function	missing functions in a file or data
Error Handling	Test of user input	Miss matches in user inputs
Boundary Related Errors	Boundaries in loop	loop miss matches, cant work in particular loop, over...
Calculation Errors	Bad logic	the bad logic data or file and cant find out a sol...
Control flow Errors	missing/wrong default	missing of default usages of data,wrong defaults

V. PREDICTION FOR REDUCTION ORDERS

In light of Section 3.1, given an occasion choice calculation IS and an element determination calculation FS, FS -> IS and IS -> FS are seen as two requests for applying diminishing procedures. Consequently, a test is the manner by which to decide the request of diminishment strategies, i.e., how to pick one between FS -> IS and IS -> FS. We allude to this issue as the expectation for diminishment orders.

i. Reduction Orders

To apply the information diminishment to each new bug informational index, we need to check the precision of both two requests (FS -> IS and IS -> FS) and pick a superior one. To maintain a strategic distance from the time cost of physically checking both diminishment orders, we consider foreseeing the diminishment arrange for another bug informational collection in light of chronicled informational indexes. As appeared in Fig 2.1, we change over the issue of forecast for lessening orders into a parallel arrangement issue. A bug informational index is mapped to an occurrence and the related decrease arrange (either FS -> IS or IS -> FS) is mapped to the name of a class of occurrences**.** Take note of that a classifier can be prepared just once when confronting numerous new bug informational indexes.

That is, preparing such a classifier once can foresee the lessening orders for all the new information sets without checking both lessening orders. To date, the issue of foreseeing decrease requests of applying highlight determination and occurrence choice has not been examined in other application situations. From the point of view of programming designing, foreseeing the diminishment arrange for bug informational indexes. Ventures of foreseeing lessening orders for bug triage. a sort of programming measurements, which includes exercises for measuring some property for a bit of programming . Notwithstanding, the components in our work are separated from the bug informational collection while the components in existing work on programming measurements are for individual programming artifacts,3 e.g., a person bug report or an individual bit of code. In this paper, to keep away from questionable significations, a characteristic alludes to an extricated highlight of a bug informational collection while an element alludes to an expression of a bug report.

VI. DISCUSSION

In this paper, we propose the issue of information diminishment for bug triage to diminish the sizes of informational indexes and to progress the nature of bug reports. We utilize strategies of occurrence choice and highlight choice to decrease clamour and repetition in bug informational indexes. Be that as it may, not all the commotion and excess are expelled. The reason for this reality is that it is difficult to precisely identify clamour and repetition in certifiable applications. On one hand, due to the expansive sizes of bug vaults, there exist no sufficient names to stamp whether a bug report or a word has a place with clam or excess; then again, since all the bug reports in a bug archive are recorded in characteristic dialects, even loud and repetitive information may contain valuable data for bug settling. In our work, we propose the information decrease for bug triage. This reality is brought about by the multifaceted nature of bug triage. We clarify such multifaceted nature as takes after. Initially, in bug reports, articulations in normal dialects might be hard to plainly see; second, there exist numerous potential designers in bug vaults ; third, it is difficult to cover all the information of bugs in a product extend and even human triagers may allocate engineers by misstep. Our work can be utilized to help human triagers as opposed to supplant them. In this paper, we build a prescient model to decide the lessening request for another bug informational collection in view of chronicled bug informational collections. Qualities in this model are measurement estimations of bug informational collections, e.g., the quantity of words or the length of bug reports. No illustrative expressions of bug information sets are separated as properties. We plan to concentrate more nitty gritty traits in future work.

In our work, we tend to introduce a determination to decide the decrease request of applying case choice and highlight choice. Our work is not a perfect determination to the expectation of diminishment requests and can be seen as a stage towards the programmed expectation. We can prepare the prescient show once and anticipate diminishment orders for each new bug informational collection. The cost of such forecast is not costly, contrasted and attempting every one of the requests for bug informational collections. Another potential issue is that bug reports are most certainly not revealed in the meantime in true bug vaults. In our work, we separate characteristics of a bug informational index and consider that every one of the bugs in this informational collection are accounted for in certain days. Contrasted and the season of bug triage, the time extend of a bug informational collection can be overlooked. Along these lines, the extraction of qualities from a bug informational collection can be connected to certifiable applications.

VII. CONCLUSION

Bug triage is a costly stride of programming upkeep in both work cost and time cost. In this paper, we consolidate highlight choice with occasion determination to lessen the size of bug informational indexes and in addition enhance the information quality. To decide the request of applying occasion choice and highlight determination for another bug informational collection, we extricate qualities of every bug informational collection and prepare a prescient model in light of chronicled informational collections. We experimentally explore the information diminishment for bug triage in bug archives of two substantial open source ventures, specifically Eclipse and Mozilla. Our work gives a way to deal with utilizing methods on information preparing to shape decreased and astounding bug information in programming improvement and upkeep. In future work, we anticipate enhancing the aftereffects of information lessening in bug triage to investigate how to set up a high quality bug informational collection and handle a space particular programming assignment. A set of users can upload files after registering into the site. Developers who are inactive will be deactivated by the classifier. Developers resolves the bug reports by specifying the solution in the specified area. New bugs will be added into the bug data set by the classifier[10]. For foreseeing lessening orders, we plan to pay endeavours to discover the potential relationship between the qualities of bug informational indexes and the lessening orders.

REFERENCES

[1] J. Anvik, L. Hiew, and G. C. Murphy, "Who should fix this bug?"in Proc. 28th Int. Conf. Softw, pp. 361–370 ,Eng., May 2006.

[2] S. Breu, R. Premraj, J. Sillito, and T. Zimmermann, "Information needs in bug reports: Improving cooperation between developers and users," in Proc. ACM Conf. Comput. Supported Cooperative Work, pp. 301–310,Feb. 2010.

[3] D. _Cubrani_c and G. C. Murphy, "Automatic bug triage using text categorization," in Proc. 16th Int. Conf. Softw. Eng. Knowl. Eng , pp. 92- 97,Jun. 2004.

[4] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," J. Mach. Learn. Res., vol. 3, pp. 1157–1182, 2003.

[5] Q. Hong, S. Kim, S. C. Cheung, and C. Bird, "Understanding a developer social network and its evolution," in Proc. 27th IEEE Int. Conf. Softw. Maintenance, pp. 323–332, Sep. 2011.

[6] J. Han, M. Kamber, and J. Pei, Data Mining: Concepts and Techniques,3rd ed. Burlington, MA, USA: Morgan Kaufmann, 2011.

[7] G. Jeong, S. Kim, and T. Zimmermann, "Improving bug triage with tossing graphs," in Proc. Joint Meeting 12th Eur. Softw. Eng.Conf. 17th ACM SIGSOFT Symp. Found. Softw. Eng.,pp. 111–120, Aug. 2009 .

[8] T. M. Khoshgoftaar, K. Gao, and N. Seliya, "Attribute selection and imbalanced data: Problems in software defect prediction,"in Proc. 22nd IEEE Int. Conf. Tools Artif. Intell.,pp. 137–144, Oct. 2010.

[9] T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen, and K.Torkkola, "LVQ_PAK: The learning vector quantization program package," Helsinki Univ. Technol., Esbo, Finland, Tech.Rep. A30, 1996.

[10] S. Kim, K. Pan, E. J. Whitehead, Jr., "Memories of bug fixes," in Proc. ACM SIGSOFT Int. Symp. Found. Softw. Eng., 2006, pp. 35–45.278 IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 27, NO. 1, JANUARY 2015

[11] D. Matter, A. Kuhn, and O. Nierstrasz, "Assigning bug reports using a vocabulary-based expertise model of developers," in Proc.6th Int. Working Conf. Mining Softw. Repositories,pp. 131–140, May 2009.

[12] E. Murphy-Hill, T. Zimmermann, C. Bird, and N. Nagappan, "The design of bug fixes," in Proc. Int. Conf. Softw. Eng., pp. 332–341, 2013.

[13] J. W. Park, M. W. Lee, J. Kim, S. W. Hwang, and S. Kim,"Costriage: A cost-aware triage algorithm for bug reporting systems,"in Proc. 25th Conf. Artif. Intell., pp. 139–144, Aug. 2011.

[14] J. C. Riquelme, J. S. Aguilar-Ru_ız, and M. Toro, "Finding representative patterns with ordered projections," Pattern Recognit., vol. 36,pp. 1009– 1018, 2003.

[15] M. Robnik-_Sikonja and I. Kononenko,"Theoretical and empirical analysis of relieff and rrelieff," Mach. Learn., vol. 53, no. 1/2,pp. 23–69, Oct. 2003.

[16] M. Rogati and Y. Yang, "High-performing feature selection for text classification," in Proc. 11th Int. Conf. Inform. Knowl. Manag., pp. 659– 661,Nov. 2002.

[17] Q. Shao, Y. Chen, S. Tao, X. Yan, and N. Anerousis, "Efficient ticket routing by resolution sequence mining," in Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining,pp. 605–613, Aug. 2008.

[18] R. J. Sandusky, L. Gasser, and G. Ripoche, "Bug report networks:Varieties, strategies, and impacts in an F/OSS development community," in Proc. 1st Intl. Workshop Mining Softw. Repositories, pp. 80–84,May 2004.

[19] E. Shihab, A. Ihara, Y. Kamei, W. M. Ibrahim, M. Ohira, B. Adams,A. E. Hassan, and K. Matsumoto, "Predicting re-opened bugs: A case study on the eclipse project," in Proc. 17th Working Conf.Reverse Eng., pp. 249–258, Oct. 2010.

[20] C. Sun, D. Lo, S. C. Khoo, and J. Jiang, "Towards more accurate retrieval of duplicate bug reports," in Proc. 26th IEEE/ACM Int.Conf. Automated Softw. Eng., pp. 253–262, 2011.

[21] R. E. Walpole, R. H. Myers, S. L. Myers, and K.Ye, Probability & Statistics for Engineers & Scientists, 8th ed. Upper Saddle River, NJ, USA: Pearson Education, 2006.

[22] X. Wang, L. Zhang, T. Xie, J. Anvik, and J. Sun, " An approach to detecting duplicate bug reports using natural language and execution information," in Proc. 30th Int. Conf. Softw. Eng.pp. 461–470, May 2008.

[23] J. Xuan, H. Jiang, Z. Ren, J. Yan, and Z. Luo,"Automatic bug triage using semi-supervised text classification," in Proc. 22nd Int. Conf. Softw. Eng. Knowl. Eng. pp. 209–214, Jul. 2010.

[24] J. Xuan, H. Jiang, Z. Ren, and W. Zou, "Developerprioritization in bug repositories," in Proc. 34th Int.Conf. Softw. Eng., 2012, pp. 25– 35.

[25] W. Zou, Y. Hu, J. Xuan, and H. Jiang, "Towards training set reduction for bug triage," in Proc. 35th Annu. IEEE Int. Comput. Soft. Appl. Conf., pp. 576–581, Jul. 2011.

[26] T. Zimmermann, N. Nagappan, P. J. Guo, and B. Murphy, "Characterizing and predicting which bugs get reopened," in Proc. 34th Int. Conf. Softw. Eng., pp. 1074–1083,Jun. 2012.

[27] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schr€oter, and C. Weiss, "What makes a goodbug report?" IEEE Trans. Softw. Eng., vol. 36, no 5, pp. 618–643, Oct. 2010.

[28] H. Zhang and G. Sun, "Optimal reference subset selection for nearest neighbor classification by tabusearch," Pattern Recognit., vol. 35 pp. 1481–1490, 2002.

[29] X. Zhu and X. Wu, "Cost-constrained data acquisition for intelligent data preparation," IEEE Trans. Knowl. Data Eng., vol. 17, no. 11, pp. 1542–1556, Nov. 2005.

[30] "Towards Effective Bug Triage with Software data Reduction Techniques," IEEETransaction On Knowledge And Data Engineering, vol. 27, no. 1,Jan. 2015.