Trends in Engineering and Technology (NCTET-2K17)
International Journal of Advanced Engineering Research and Science (IJAERS) *Special Issue-3*
https://dx.doi.org/10.22161/ijaers/nctet.2017.55 *ISSN: 2349-6495(P) | 2456-1908(O)*

An Efficient improved technique to retrieve the bug repository

P. Hari Prasad
Assistant Professor
Dept of CSE
Sree Vidyanikethan Engineering College
Tirupati
prasadghari@gmail.com

D. Ganesh
Assistant Professor
Dept of CSE
Sree Vidyanikethan Engineering College
Tirupati
dgani05@gmail.com

Dr. M.Sunil Kumar
Professor & HOD
Dept of CSE,
Sree Vidyanikethan Engineering College
Tirupati
Sunilmalchi1@gmail.com

ABSTRACT

Bug triage is the most important step in handling the bugs which occur during a software process. In manual bug triaging process the received bug is assigned to a tester or a developer by a triager, hence the bugs are received in huge numbers it is difficult to carry out the manual bug triaging process, and it consumes much resources both in the form of man hours and economy, hence there is a necessity to reduce the exploitation of resources. Hence a mechanism is proposed which facilitates much better and efficient triaging process by reducing the size of the bug data sets, the mechanism here involves techniques like clustering techniques and selection techniques, The approach proved much efficient than the manual bug triaging process when compared with bug data sets which were retrieved from the open source bug repository called bugzilla.

INTRODUCTION

Most of the software industries spend nearly half of their economy, that is more than 45 percent of their economy in finding out and correcting bugs alone, this is occurring due to the exploitation of resources both in the form of human labor and economy by implementing manual bug triaging process, here the bug are manually allocated to the testers or developers by the triagers, since the incoming bugs are all over from the world it is a tedious process and require much man power and economy to facilitate the process.

So, there is a necessity to reduce the expenditure, to deal with this exploitation previously few techniques have been proposed like text classification techniques, Naïve bayes, Support vector machine which facilitated the automation of the process, and also selection mechanisms like feature selection and instance selection have been used to reduce the size of the bug data sets for a better triaging process.

As few researchers proposed that triaging produces better results when the data undergoes preprocessing before any machine learning algorithms are applied to it, Hence a combination of clustering mechanism and selection mechanism is used for a better triaging process, primarily the data is prepared by pruning the data with the help of techniques like tokenization, stop words, stemming etc.., and an algorithm called x means algorithm is used to cluster the data, and two selection mechanism called instance selection and feature selection are used to reduce the size of the bug data set and provide better triaging process.

ITERATURE SURVEY

Software engineering data contains useful information such as code bases, execution traces, bug databases etc.., by exploiting such data we can build a more reliable software system by building dependencies based on the historical data produced in the software process. [2] A.E Hassan

https://dx.doi.org/10.22161/ijaers/nctet.2017.55 *ISSN: 2349-6495(P) | 2456-1908(O)*

stated that most commonly available repositories of a software system are source control repositories, bug repositories, archived communications, deployment logs and code repositories. The mining software repositories concept which is called as MSR primarily analyzes and cross references all the data which is present in the repository to extract the data which is considered to be useful in the repository, by evolving these repositories from basic repositories which contain records to dynamic repositories which can be used for decision making in the aspect of software engineering. [11] John anvik stated that open source projects generally support open repositories in which both the user and the developers can access and report bugs. The reports which are received in the repository should undergo the process of triaging to determine whether the report is worthy or not, if it is that particular report should be assigned to a developer for further rectification, Hence in the context of large data repositories the

[4] Davor C Urbanic stated that, a bug tracking system is necessary to maintain bug reports on large software development projects.

Research has been carried out by the researchers in the field of mining software repositories which augment traditional software engineering information by the use of tools and techniques, which are implemented in solving challenging problems in the field of software which the engineers face on a daily during the process.

Triage is actually a medical term in which the patients are allocated based on their severity, in a similar way bug triage is a term in which the bugs are allocated to the testers based on their priority.

[14]Silvia Breu stated that, in open source projects such as Mozilla, bugzilla and eclipse, the bug tracking systems interact with the user communities, so that the users can be a part of bug fixing process, hence this type of bug tracking systems play an important role in interacting and communicating with the users. [5] Dominuque Matter stated that, the daily incoming of bug reports is high on daily basis, hence triaging these incoming reports is necessary and it is a difficult task assigning a developer to a report is also a part of it. Traditional bug consuming is a very tedious and time consuming job, the repositories which contain the bugs are called bug repositories, bugs are collected into a data base, and the triaging team sort out the bugs based on their priority.

[7]Gaeul Jeong stated that, It is important that the bugs should be identified and rectified within time in software engineering process, Most of the bugs are manually allocated by human triagers which is a tedious task. [12] Nicolas Bettenburg stated that bug reports acts as the main source of information for developers, but there exists a difference in the quality of each bug report, on the open source repositories like apache, eclipse and Mozilla a survey has been conducted with the help of the users and developers, in that survey more than 450 responses mismatched with the user requirement, to overcome this methods like reproducing, stack traces are used which are difficult to maintain for large number of bug reports.

[13] Shivkumar Shivaji stated that prediction of bugs from source code files are done by using machine learning classifiers, primarily the classifier is linked with historical bug data and used for prediction the possible outcomes, yet these classifiers suffer from a drawback that is insufficient performance real world situations and slow prediction time, some of the classifiers are Naïve Bayes and support vector machine (SVM). Feature selection can defined as the process in which the irrelevant features are deducted and detecting only the relevant ones, an optimal selection of features can bring improvement in overall knowledge of domain, reduced size, generalization capacity etc.., [9] J.Arturo Olvera Lopez stated that sufficient identification of features is necessary in real world scenario, hence the identification of features is important. [20] Yiming Yang stated that, feature selection is the best solution for text classification problems it increases both the classification effectiveness and also computational efficiency. Instance selection is a process, in which the dataset size is reduced , which eventually decreases the runtime, especially in the case of instance based classifiers, the commonly used instance selection mechanisms are wrapper and filter, here

https://dx.doi.org/10.22161/ijaers/nctet.2017.55 *ISSN: 2349-6495(P) | 2456-1908(O)*

filtering mechanism is used called as Iterative case filtering(ICF) algorithm.

[9] J. Arturo Ovlera Lopez stated that historical data that is previously known data is used to classify new instances.

PROBLEM STATEMENT

In software development process, the output of the software process is stored in large data bases called software repositories. Conventional method of software analysis is not suitable for large databases, hence there is a necessity to reduce the size of the bug data. Data reduction for bug triage aims to build a small scale and efficient database by removing uninformative data, the problem of data reduction for bug triaging is how to reduce the size of the bug data to reduce labor cost of the developers and eventually increase the process of bug triaging.

It primarily concentrates on the problem of data reduction for bug triage in two aspects, namely

i) Reduce the scale of bug dimension
ii) Improve the accuracy of the bug triaging.

PROPOSED METHODOLOGY

The proposed methodology starts by taking the bug data sets from open source bug repositories like Bugzilla, Eclipse.

FRAMEWORK:

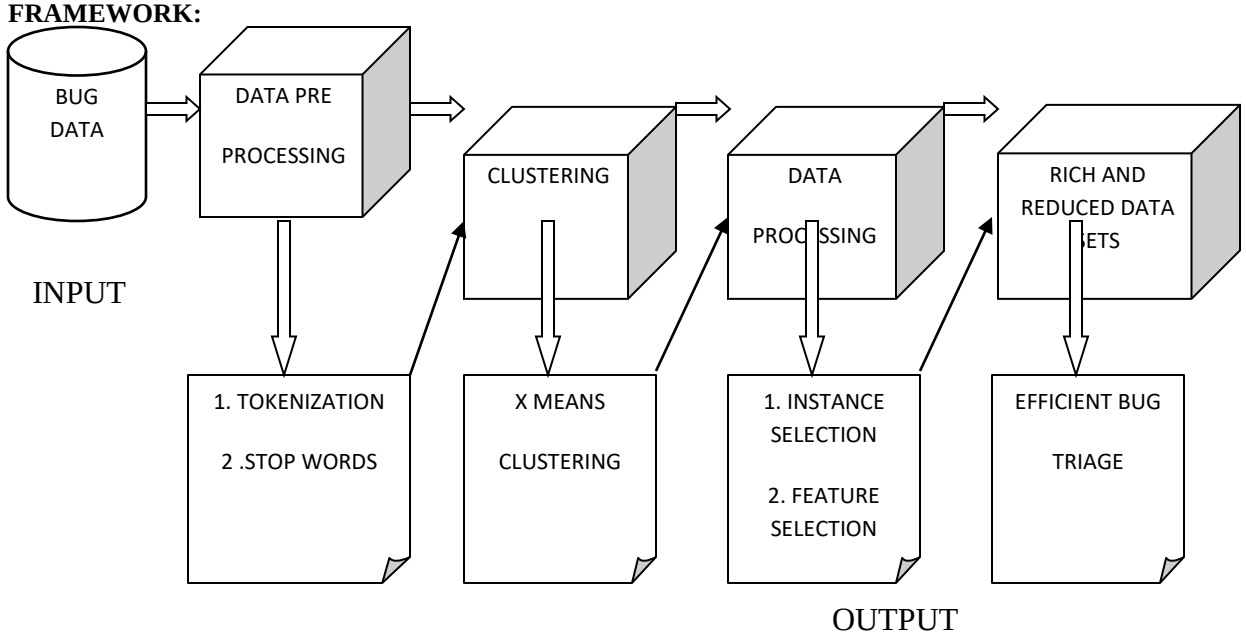


Figure1. Framework for an efficient bug triage using clustering mechanism

Initially the input data is taken as data sets and then the data is preprocessed using preprocessing mechanisms and the output of the preprocessing state is pruned data, and the pruned data is taken as input and then clustering mechanisms are used and the pruned data is clustered, and the clustered data is taken as input and selection mechanisms are applied and the output of selection mechanism is reduced data sets which is further used for bug triaging.

According to the frame work, the downloaded bug data sets are prepared by preprocessing mechanisms, the preprocessing mechanism which are being used are

a. Tokenization:

   Tokenization can be defined as the pre processing mechanism in which the given data is broken down into tokens or characters.

b. Stop words:

   Stop word can be defined as the pre processing mechanism in which the grammar content that is

https://dx.doi.org/10.22161/ijaers/nctet.2017.55 *ISSN: 2349-6495(P) | 2456-1908(O)*

removing capitalized words, punctuation marks and conjunctions like, the, it, this, there, us, we etc..,

On the completion of the preprocessing of bug data sets, the bug data sets are further processed by an algorithm called x means clustering algorithm.

ALGORITHM 1: X means clustering algorithm

Step 1: Initialize K= K min

Step 2: Run k means algorithm

Step 3: for k=1,….. K:Replace each centroid by two centroids μ_1 and μ_2.

Step 4: Run k means algorithm with k =2 over the cluster k, replace or retain each centroid based on model selection criterion.

Step 5: if convergence condition is not satisfied, go to step 2, else exit.

Here in this algorithm model selection criterion is a condition called Bayesian Information criterion (BIC), the algorithm performs a model selection test using BIC to decide wether the newly formed clusters are better than the original clusters or not,

BIC {k} = -2* logL + k logN.

Where N is the observations and k is the number of clusters and logL is the log likelihood,

The following step after the x means clustering is selection mechanisms which are instance selection and Feature selection,

Instance selection is a selection mechanism which uses the instance in which the bug occurred as the criteria, it is used to remove the noisy data, the algorithm which is used for instance selection is Automatic branch and bound algorithm(ABB).

ALGORITHM 2: Automatic branch and bound algorithm

Input:

S(X) = A sample space

J= Evaluation measure

Output:

L = All equivalent solutions found

procedure ABB(S(X); sample ; var L` list of set)

for each x in X do

enqueue(Q,X\{x}) // remove a feature at a time

end

while not empty(Q) do

X`:= dequeue(Q) // X` is legitimate if it is not a subset of a pruned state

If legitimate(X`) and J(S(X`)) $\geq$ J_0 then

L`:= append (L`, X`)

ABB(S(X`), L`)

end

end

end

begin

Q:= Ø // Queue of pending states

L`:= [X] // List of solutions

J_0:= J(S(X)) // Minimum allowed value of J

ABB(S(X), L`) // Initial call to ABB

k:= smallest size of subset in L`

L := set of elements of L` of size k

end

Here Initially a Procedure is created ABB S(X), where S(X) is a sample space, x is derivation of sample space var L` is the list set, for each subset x is present in X then the following steps should be done they are,

Step 1: en queue (Q, X\[{x}) here {x} is a random subset of a feature set, in this step each feature set is removed from the sample space and inserted into the queue.

Step 2: By using a while loop the queue is checked whether it is empty or not if it is empty we create a new sample space called as X` and we de queue all the elements from the queue Q into the other sample space X`.

Step 3: next if the data present in the sample space X` is legitimate, that is if it not a subset of pruned state, then by

https://dx.doi.org/10.22161/ijaers/nctet.2017.55 *ISSN: 2349-6495(P) | 2456-1908(O)*

using an 'if' statement we $J(S(X`)) \geq J_0$ where as $J(S(X`))$ is an object function and J_0 is the minimum allowed value.

Step 4: Then L`:= append (L`, X`), where as L` is a newly formed list of elements and the feature sets present in the sample space X` are appended into the newly formed list L`

Step 5: Finally the procedure is closed by appending the list L` value sto the derived sample space S(X`).

By using this algorithm the noisy or uninformative data present in the data sets is reduced, In this context the uninformative clustered data is removed from the dataset.

Feature selection is a selection mechanism which uses the features exhibited as the criteria, it is used to reduce redundancy, the algorithm which is used for feature selection is called as Iterative case filtering algorithm (ICF).

ALORITHM 3: Iterative case filtering algorithm (ICF)

for all x ∈ T do
if x classified incorrectly by k nearest neighbor then
flag x for removal
for all x ∈ T do
compute reachable(x)
compute coverage(x)
progress = false
for all x ∈ T do
if |reachable(x) > |coverage(x)| then
flag x for removal
progress = true
for all x ∈ T do
if x flagged for removal then T = T-{x}
until not progress
return T

Step 1: Here by using a condition for all x ∈ T where as 'x' is the subset and 'T' is the training set, it is checked whether the subset belongs to training set, if the condition is satisfied then by 'if' condition 'x' value is checked whether it is classified correctly or incorrectly this is done by using k nearest neighbor algorithm then flag 'x' for removal.

Step 2: Then again check for all x ∈ T, if 'x' is flagged then T=T-{x}, this means that the subset {x} should be removed from the training set 'T'.

Step 3: Iterate the above steps until no flagged items are left in the training set.

Step 4: Again check the condition for all x ∈ T, do compute reach ability (x) , compute coverage(x).

Step 5: for all x ∈ T, do if |reach ability(x)| > |coverage| then flag 'x' for removal.

Step 6: for all x ∈ T, do if 'x' is flagged for removal, then T=T-{x}, that is remove the subset from the training set, Return T.

All these mechanism are applied on bug datasets which are used from the open source repository, a sample bug data set is given below.

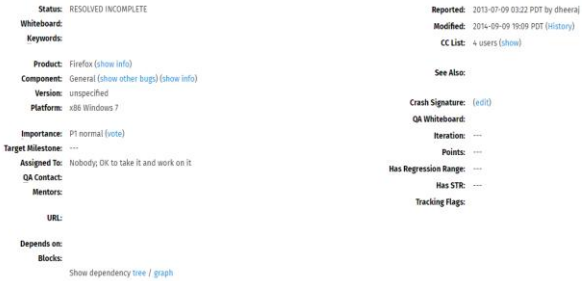


Fig.1. Sample bug dataset from an open source bug repository regarding a bug(891298) called 'error'.

Here the primary attributes of a bug dataset are

#Bug id = The allocation number of a particular bug id.

#Product = The product in which the bug occurred.

#Status = The present status of bug.

#Component= The component in which the bug occurred.

#Version= Version of the product.

#Platform= Platform on which the product is being used.

#Importance= The level of prioritization of bug.

#Assigned to= The developer or the tester to whom the bug is assigned.

#Reported= Details about the time and date and the person who reported the bug.

https://dx.doi.org/10.22161/ijaers/nctet.2017.55 *ISSN: 2349-6495(P) | 2456-1908(O)*

#Modified= Details about the time and date and the person who modified the bug.

As previous research done by Jifen Xuan et al., on bug datasets from open source repositories like Mozilla and Eclipse, The evaluation of data reduction can be calculated by comparing two aspects they are, accuracy of bug triage and the scale of bug data.

Based on those aspects a mathematical model can be developed it is,

$$Accuracy = \frac{correctly\ assigned\ bug\ reports\ in\ k\ candidates}{all\ bug\ reports\ in\ the\ test\ set}$$

Based on this mathematical method, Jifen Xuan et al., compared ten datasets five in Mozilla and five in eclipse, it is given below.

Name	DS E1	DS E2	DS E3	DS E4	DS E5
Range of bug	200001-220000	220001-240000	240001-260000	260001-280000	280001-300000
Bug reports	11,313	11,788	11,495	11,401	10,404
Words	38,650	39,495	38,743	38,772	39,333
Developers	266	266	286	260	256

Table.1. Datasets of eclipse.

Name	DS M1	DS M2	DS M3	DS M4	DS M5
Range of bug	400001-440000	440001-480000	480001-520000	520001-560000	560001-600000
Bug reports	14,659	14,746	16,479	15,483	17,501
Words	39,749	39,113	39,610	40,148	41,557
Developers	202	211	239	242	273

Table.2. Datasets of Mozilla.

These are the previous readings which are taken by comparing different instance selection anf feature selection mechanisms.

But here 5 different data sets have been used which are named as BDS1, BDS2, BDS3, BDS4, BDS5 which are extracted from open source bug repository called bugzilla. Here BDS represents bug data sets, Each bug data set has different number of bug reports in it, all the bug data sets that is from BDS1 to BDS 5 are approximately ranged from 190900 to 1277000, The details of all the considered bug data sets are given below in a tabular form.

Name	BDS1	BDS2	BDS13	BDS14	BDS5
Range	1301049 - 1272508	23565-1274810	396690 - 1274810	190963 - 1273153	275671 - 841549
Bug reports	100	125	150	175	200
Words	20,096	2,561	3,219	3,846	4,364

Table 3. Experimental sample which are considered (bugzilla).

The experimental results obtained after the procedure of the experiment are tabulated in the give table.

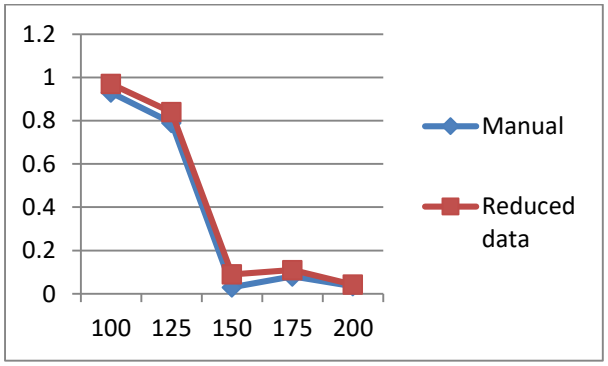
List	Name	Range	Bug reports	words	Accuracy (Manual)	Accuracy (Reduced data)
1	BDS1	1301049-1272508	100	20,096	0.93	0.97
2	BDS	23565-	125	2,561	0.79	0.81

https://dx.doi.org/10.22161/ijaers/nctet.2017.55 *ISSN: 2349-6495(P) | 2456-1908(O)*

	2	1274810				
3	BDS 3	396690 - 1274810	150	3,219	0.03	0.07
4	BDS 4	190963 - 1273153	175	3,846	0.08	0.11
5	BDS 5	275671 - 841549	200	4,364	0.035	0.042

Table 5. Comparison between manual and reduced data triaging.

The above comparison of five different data sets ranging from BDS1 to BDS 5, by using bothe the manual triaging and by triaging after data is reduced is shown graphically below.



Graph 1. Comparision between manual triaging and reduced data triaging.

4. CONCLUSION

Bug triage is an expensive step in software maintenance in both cost of labor and time, here the combinations of different selection techniques such as instance selection and feature selection are used to reduce the scale of bug data sets and also improve the quality of the bug datasets. This process is further enhanced by using clustering mechanism like x means clustering as a pre processing mechanism, Due to this approach the accuracy of the triaging process is increased when compared to manual bug triage and the bug triage using selection mechanisms. For future work planning can be done in the way of data reduction by implementing further techniques to reduce size of data sets and facilitate a better bug triaging process.

REFERENCES

[1] Ahmed E. Hassan, "Mining software engineering data", ICSE "10, May 2-8 2010, Cape Town, South Africa, Copyright 2010 ACM 978-1-60558-719-6/10/05.

[2] Ahmed E. Hassan, "The Road Ahead for Mining Software Repositories", Software Analysis and Intelligence Lab (SAIL), School of Computing, Queen's University, Canada.

[3] B. Fitzgerald, "The transformation of open source software", MIS Quart., vol. 30, no. 3, pp. 587–598, Sep. 2006.

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

[5] 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, May 2009, pp. 131–140.

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

[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., Aug. 2009, pp. 111–120.

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

[9] J. A. Olvera-L_opez, J. A.Carrasco-Ochoa, J. F. Mart_ınez-Trinidad, and J. Kittler, "A review of instance selection methods," Artif. Intell. Rev., vol. 34, no. 2, pp. 133–143, 2010.

https://dx.doi.org/10.22161/ijaers/nctet.2017.55 *ISSN: 2349-6495(P) | 2456-1908(O)*

[10]J. Anvik and G. C. Murphy, "Reducing the effort of bug report triage: Recommenders for development-oriented decisions," ACM Trans. Soft. Eng. Methodol., vol. 20, no. 3, article 10, Aug. 2011.

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

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

[13] S. Shivaji, E. J. Whitehead, Jr., R. Akella, and S. Kim, "Reducing features to improve code change based bug prediction," IEEE Trans. Soft. Eng., vol. 39, no. 4, pp. 552–569, Apr. 2013.

[14] 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, Feb. 2010, pp. 301–310.

[15] S. Kim, H. Zhang, R. Wu, and L. Gong, "Dealing with noise in defect prediction," in Proc. 32nd ACM/IEEE Int. Conf. Softw. Eng., May 2010, pp. 481–490.

[16] 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.

[17] Tommaso Dal Sasso, Michele Lanza ,"in*Bug: Visual Analytics of Bug Repositories", REVEAL @ Faculty of Informatics University of Lugano, Switzerland.

[18] V. Cerver_on and F. J. Ferri, "Another move toward the minimum consistent subset: A tabu search approach to the condensed nearest neighbor rule," IEEE Trans. Syst., Man, Cybern., Part B, Cybern., vol. 31, no. 3, pp. 408–413, Jun. 2001.

[19] Y. Fu, X. Zhu, and B. Li, "A survey on instance selection for active learning," Knowl. Inform. Syst., vol. 35, no. 2, pp. 249–283, 2013.

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

[21] 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., Jul. 2010, pp. 209–214.