International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

COPY RIGHT

2020 IJIEMR.Personal use of this material is permitted. Permission from IJIEMR must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. No Reprint should be done to this paper, all copy right is authenticated to Paper Authors

IJIEMR Transactions, online available on 30th June 2020. Link
:http://www.ijiemr.org/downloads.php?vol=Volume-09&issue=ISSUE-06

Title: IMPROVING BUG TRIAGING WITH HIGH CONFIDENCE PREDICTIONS AT ERICSSON
Volume 09, Issue 06, Pages: 181-186

Paper Authors

GURRALA PRAKASH, C.SIVA

USE THIS BARCODE TO ACCESS YOUR ONLINE PAPER

To Secure Your Paper As Per UGC Guidelines We Are Providing A Electronic Bar Code

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

IMPROVING BUG TRIAGING WITH HIGH CONFIDENCE PREDICTIONS AT ERICSSON

GURRALA PRAKASH, C.SIVA

PG SCHOLAR.DEPT OF CSE, SIR C.V. RAMAN INSTITUTE OF TECHNOLOGY & SCIENCE, AP, INDIA
ASSOCIATE PROFESSOR, DEPT OF CSE, SIR C.V. RAMAN INSTITUTE OF TECHNOLOGY & SCIENCE,, AP, INDIA

ABSTRACT: Bug triaging refers to the process of assigning a bug to the most appropriate developer to fix. It becomes more and more difficult and complicated as the size of software and the number of developers increase. In this paper, we propose a new framework for bug triaging, which maps the words in the bug reports (i.e., the term space) to their corresponding topics (i.e., the topic space). We propose a specialized topic modeling algorithm named multi-feature topic model (MTM) which extends Latent Dirichlet Allocation (LDA) for bug triaging. MTM considers product and component information of bug reports to map the term space to the topic space. Finally, we propose an incremental learning method named TopicMiner which considers the topic distribution of a new bug report to assign an appropriate fixer based on the affinity of the fixer to the topics. We pair Topic Miner with MTM (TopicMiner MTM). We have evaluated our solution on 5 large bug report datasets including GCC, Open Office, Mozilla, Netbeans, and Eclipse containing a total of 227,278 bug reports. We show that Topic Miner MTM can achieve top-1 and top-5 prediction accuracies of 0.4831 - 0.6868, and 0.7686 - 0.9084, respectively. We also compare TopicMinerMTM with Bugzie, LDA-KL, SVM-LDA, LDA-Activity, and Yang et al.'s approach. The results show that TopicMinerMTM on average improves top-1 and top-5 prediction accuracies of Bugzie by 128.48% and 53.22%, LDA-KL by 262.91% and 105.97%, SVM-LDA by 205.89% and 110.48%, LDA-Activity by 377.60% and 176.32%, and Yang et al.'s approach by 59.88% and 13.70%, respectively.

1.INTRODUCTION

Bugs show up during programming advancement and upkeep, and bug fixing is a tedious and expensive undertaking. Numerous product ventures use bug following frameworks (e.g., Bugzilla and JIRA) to oversee bug revealing, bug goals, and bug documenting forms [9]. Beside bug depiction and synopsis data, a run of the mill bug report records different sorts of helpful data, e.g., item and part. We allude to this data as highlights of a bug report. Figure 1 presents a bug report from Eclipse with BugID=212000.1 In the figure, we notice that the bug report has a place with item CDT and part cdt-center. When a bug report is gotten, allocating it to an appropriate engineer inside a brief timeframe span can diminish the time and cost of the bug fixing process. This task procedure is known as

bug triaging (e.g., in Figure 1, the bug is alloted to Oleg Krasilnikov2). Bug triaging is a tedious procedure since frequently numerous designers are engaged with programming advancement and support. For Eclipse and Mozilla, in excess of 1,800 designers took an interest in the bug fixing process (see Table 2). On the off chance that the entirety of the bug reports should be physically allocated to the most proper designers, the bug triaging errands would take a ton of time and exertion. To help in finding fitting engineers, programmed bug triaging approaches have been proposed [7], [10], [20], [38]. Huge numbers of these methodologies utilize the vector space model (VSM) to speak to a bug report, i.e., a bug report is treated as a vector of terms (words) and their checks. Notwithstanding, designers frequently utilize different terms to communicate a similar significance. A similar term can likewise convey various implications relying upon the unique circumstance. These equivalent and polysemous words can't be caught by VSM.

In the data recovery network, theme displaying [36], which can gather the inalienable idle subjects of a printed record, has been utilized as an approach to manage equivalent words and polysemy issues. A point model proselytes terms in an archive to subjects. Two terms that are distinctive would now be able to be esteemed comparative on the off chance that they are of a similar subject which tends to the equivalent word and polysemy issues. Different point demonstrating calculations are proposed in the writing including Latent Semantic Indexing/Analysis (LSA) [16], probabilistic LSA (pLSA) [18], and Latent Dirichlet Allocation (LDA) [12].

Among the three, LDA is the most as of late proposed and it tends to the constraints of LSA and pLSA [12]. LDA thinks about an archive as an irregular blend of idle subjects, where a theme is an arbitrary blend of terms. We expand LDA and propose another point model named multi-include theme model (MTM) for the bug triaging issue. Since a bug report has different highlights (e.g., item influenced by the bug, part influenced by the bug, and so on.), MTM considers the highlights of a bug report when it changes over terms in the printed portrayal of the report (i.e., messages in the synopsis and depiction fields of the report) to their relating subjects in the point space. Given a bug report with a specific element mix (i.e., item part blend), MTM changes over a word in the bug report, to a subject. Like standard theme displaying calculation, as Latent Dirichlet Allocation (LDA) [12], the word to point change is finished by taking a gander at cooccurrences of words in records (for our situation: bug reports outlines and portrayals). In any case, unique in relation to LDA, when changing over words to subjects in a bug report with a specific element mix, MTM puts an uncommon accentuation on the appearances of words in bug reports with a similar element blend, without disregarding the word appearances in all other bug reports. Since the quantity of bug reports of a specific element mix is frequently restricted, to surmise better points, MTM needs to likewise consider

terms that show up in bug reports having a place with other component blends. MTM thinks about every mix of highlights as an irregular blend of idle themes, where a subject is an arbitrary blend of terms. MTM is an extensible point model, where at least one highlights can be thought about. We allude to an element as an absolute field in a bug report that a bug journalist can fill when the columnist presents a bug report. These fields incorporate the item, part, columnist, need, seriousness, OS, adaptation, and stage fields. We prohibit the regular language depictions in the bug reports, which incorporates the substance of the outline and portrayal fields, as the highlights since they are not absolute in nature. In this paper, we utilize the item segment mix as the information include blend, since item and part are two of the most significant highlights that depict a bug. Given a bug report with a specific element mix, MTM changes over a term in the bug report to a point by putting uncommon accentuation on the appearances of the word in bug reports with a similar component blend, without overlooking the word appearances in all other bug reports.

We propose another methodology for bug triaging which influences MTM. We take as information a preparation set of bug reports (whose fixers are known) and another bug report whose fixer is to be anticipated. Our methodology, named TopicMinerMTM processes the liking of an engineer to another bug report, in view of the reports that the designer fixed previously. To do this, we analyze the subjects that show up in the new bug report with those in the old reports that the engineer has fixed previously.

2.EXISTING SYSTEM:

To aid in finding appropriate developers, automatic bug triaging approaches have been proposed in the existing. Many of these approaches use the vector space model (VSM) to represent a bug report, i.e., a bug report is treated as a vector of terms (words) and their counts. However, developers often use various terms to express the same meaning. The same term can also carry different meanings depending on the context. These synonymous and polysemous words cannot be captured by VSM.

Various topic modeling algorithms are proposed in the literature including Latent Semantic Indexing/Analysis (LSA), probabilistic LSA (pLSA), and Latent Dirichlet Allocation (LDA). Among the three, LDA is the most recently proposed and it addresses the limitations of LSA and pLSA.

DISADVANTAGES OF EXISTING SYSTEM:

LDA considers a document as a random mixture of latent topics, where a topic is a random mixture of terms.

One or few features can be only taken into consideration.

Lower accuracy.

More complex

More time taken

3.PROPOSED SYSTEM:

We extend LDA and propose a new topic model named multi-feature topic model (MTM) for the bug triaging problem. Since

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

a bug report has multiple features (e.g., product affected by the bug, component affected by the bug, etc.), MTM considers the features of a bug report when it converts terms in the textual description of the report (i.e., texts in the summary and description fields of the report) to their corresponding topics in the topic space. Given a bug report with a particular feature combination (i.e., product component combination), MTM converts a word in the bug report, to a topic.

We refer to a feature as a categorical field in a bug report that a bug reporter can fill when the reporter submits a bug report. These fields include the product, component, reporter, priority, severity, OS, version, and platform fields. We exclude the natural language descriptions in the bug reports, which includes the contents of the summary and description fields, as the features since they are not categorical in nature.

In this paper, we use the product-component combination as the input feature combination, since product and component are two of the most important features that describe a bug. Given a bug report with a particular feature combination, MTM converts a term in the bug report to a topic by putting special emphasis on the appearances of the word in bug reports with the same feature combination, without ignoring the word appearances in all other bug reports.

ADVANTAGES OF PROPOSED SYSTEM:

MTM considers each combination of features as a random mixture of latent topics, where a topic is a random mixture of terms.

MTM is an extensible topic model, where one or more features can be taken into consideration.

We propose a new approach for bug triaging which leverages MTM. We take as input a training set of bug reports (whose fixers are known) and a new bug report whose fixer is to be predicted.

Our approach, named TopicMiner MTM computes the affinity of a developer to a new bug report, based on the reports that the developer fixed before. To do this, we compare the topics that appear in the new bug report with those in the old reports that the developer has fixed before.

4. SYSTEM ARCHITECTURE:

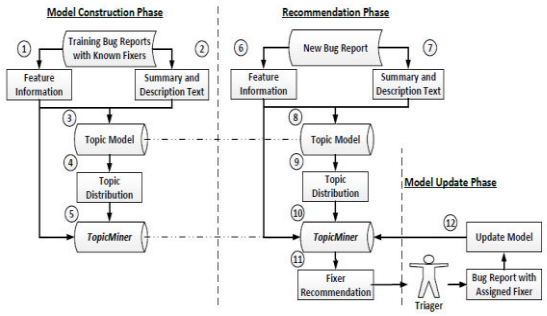

Fig 1 **SYSTEM ARCHITECTURE**

5. IMPLEMENTATION

Admin

In this module, the Admin has to login by using valid user name and password. After login successful he can do some operations such as search all bug status, view all request, request & response and top developer etc.

Request & Response

In this module, the admin can view the all the developer request and response. Here all the request and response will be stored with their tags such as Id, requested user photo, requested user name, user name request to, status and time & date. If the user accepts the request then status is accepted or else the status is waiting.

Developer

In this module, there are n numbers of users are present. User should register before doing some operations. And register user details are stored in user module. After registration successful he has to login by using authorized user name and password. Login successful he will do some operations like view or search Bug details, send request, view topic model, check inbox.

6.CONCLUSION

We propose a new topic model based bug triaging approach, named Topic Miner, and a new topic model, named multi-feature topic model (MTM), which takes into consideration the features of a bug report when assigning topics to words in the report. We have evaluated our solution on 227,278 bugre ports from five software systems and demonstrate that Topic Miner MTM outperforms Bugzie, LDA-KL, SVM-LDA,LDA-Activity, and Yang et al.'s approach by substantial margins.

In the future, we plan to improve the effectiveness of our approach further, and investigate additional bug reports. Also, in this work, we merge the two features (i.e., product and component) as one composite feature (i.e., by creating a feature combination). Other ways of using the multiple features exist and we plan to explore them in a future work. We also plan to design a better topic model to predict fixers when the number of bug reports in a specific product component combination is small (e.g., by using a mixture of models which includes a general model that the approach can back off to when the number of bug reports in a specific product-component combination is small).

REFERENCES

[1] Eclipse bug tracking system. https://bugs.eclipse.org/bugs/.

[2] Gcc bug tracking system. http://gcc.gnu.org/bugzilla/.

[3] Mozilla bug tracking system.

[4] Netbeans bug tracking system. http://netbeans.org/bugzilla/.

[5] Openoffice bug tracking system. https://issues.apache.org/ooo/.

[6] H. Abdi. Bonferroni and ˇsid´ak corrections for multiple comparisons.in nj salkind (ed.). encyclopedia of measurement andstatistics. Encyclopedia of measurement and statistics, 2007.

[7] J. Anvik, L. Hiew, and G. Murphy. Who should fix this bug? InProceedings of the 28th international conference on Software engineering,pages 361–370, 2006.

[8] J. Anvik and G. Murphy. Determining implementation expertisefrom bug reports. In Mining Software Repositories, 2007. ICSEWorkshops MSR'07. Fourth International Workshop on, 2007.

[9] D. Bertram, A. Voida, S. Greenberg, and R. Walker. Communication,collaboration, and bugs: the social nature of issue tracking

insmall, collocated teams. In Proceedings of the 2010 ACM conferenceon Computer supported cooperative work, pages 291–300. ACM, 2010.

[10] P. Bhattacharya and I. Neamtiu. Fine-grained incremental learningand multi-feature tossing graphs to improve bug triaging. InSoftware Maintenance (ICSM), 2010 IEEE International Conference on,pages 1–10. IEEE, 2010.

[11] D. Binkley, D. Heinz, D. Lawrie, and J. Overfelt. Understandinglda in source code analysis. In Proceedings of the 22Nd InternationalConference on Program Comprehension, pages 26–36. ACM, 2014.

[12] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. Journalof Machine Learning Research, 3:993–1022, 2003.

[13] G. Bortis and A. van der Hoek. Porchlight: A tag-based approachto bug triaging. In Software Engineering (ICSE), 2013 35th InternationalConference on, pages 342–351. IEEE, 2013.

[14] N. Cliff. Ordinal methods for behavioral data analysis. PsychologyPress, 2014.

[15] D. C˘ ubranic´. Automatic bug triage using text categorization. InIn SEKE 2004: Proceedings of the Sixteenth International Conference onSoftware Engineering & Knowledge Engineering, 2004.

[16] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, andR. Harshman. Indexing by latent semantic analysis. Journal of theAmerican Society for Information Science, 41(6):391–407, 1990.

[17] G. Heinrich. Parameter estimation for text analysis. Web:http://www.arbylon.net/publications/text-est.pdf, 2005.

[18] T. Hofmann. Probabilistic latent semantic analysis. In UAI, pages289–296, 1999.