Review of Text Mining Techniques for Software Bug Localization

Conference Paper · January 2019

DOI: 10.1109/CONFLUENCE.2019.8776959

CITATIONS
3

READS
34

2 authors:

Tamanna Sharma
Guru Jambheshwar University of Science & Technology
11 PUBLICATIONS **6** CITATIONS
SEE PROFILE

Om PRAKASH Sangwan
Guru Jambheshwar University of Science & Technology
78 PUBLICATIONS **539** CITATIONS
SEE PROFILE

Review of Text Mining Techniques for Software Bug Localization

Tamanna
Dept. of computer science and engineering,
Guru JamheshwarUniversity of Science and Technology
Hisar,India
tamannasharma100@gmail.com

Om Prakash Sangwan
Dept. of computer science and engineering,
Guru JamheshwarUniversity of Science and Technology
Hisar,India
Sangwan0863@gmail.com

Abstract— **Software Bug Localization (SBL) is a task of locating the buggy source code. There are various ways of doing SBL and one of them is static SBL which utilizes the power of Text Mining (TM) in association with software repositories. Most of static SBL models are based on Information Retrieval (IR) methodology in which bug report works as a query and source code as database. In this paper we review state of the art SBL models which uses text mining techniques as their back bone in conjunction with other techniques. Essential features are extracted and summarized with the help of tabular representation. Aim of doing this study is to find the gaps in previous SBL models for proposing a novel SBL model in future.**

Keywords—Mining Software Repositories (MSR),Fault, Software Bug Localization (SBL), Information Retrieval (IR)

I. INTRODUCTION

There are various methods of doing software bug localization but after classical debugging major automated categories came in to existence were dynamic and static. Dynamic techniques deals with execution part or dependent on test case or test suites execution. While static techniques deals with static structure of program code which is independent in terms of execution traces. Software artefacts which were produced during software development life cycle are rich source of information. It supports various software development tasks like software bug prediction, bug severity prediction, software bug localization etc. Static techniques used source code, bug report, old bug repositories which is independent of execution of a particular instance. Most of them took the advantage of Information Retrieval techniques, in which bug report was taken as query and searching was accomplished on source code for the localization of bugs. But this was the basic crux of static SBLM. Lots of amendment was made and still going on to make it more accurate and Robust.

This paper is designed to show the current state of popular text mining techniques employed in static software bug localization models. Next section (section 2) gives an overview of SBL models and text mining techniques used by them. Third section explains major findings of SBL models and last section is about conclusion of text mining techniques.

II. TEXT MINING TECHNIQUES IN CONJUNCTION WITH SBL

Text mining technique is useful in Bug localization because size of source code is very large and when any bug arises it is difficult for developer to check whole source code files while only a small part of code is responsible for that particular bug and sometimes it was also possible that two or three different source code files are responsible for a bug. To reduce the effort and wastage of resources researchers took the advantage of bug report generate during the submission of bug. Text mining technique is employed between bug report attributes like summary, description etc. and source code because of the similarity present between them. As shown in fig.1. different text mining techniques was employed by previous SBLM's.

Topic Modelling is proved to be one of the promising areas for localization of bugs. Lukins *et.al.* explored all possibilities of LDA for SBL and shows significant improvement in comparison with LSI and PLSI [1]. Because LDA is a fully generative and

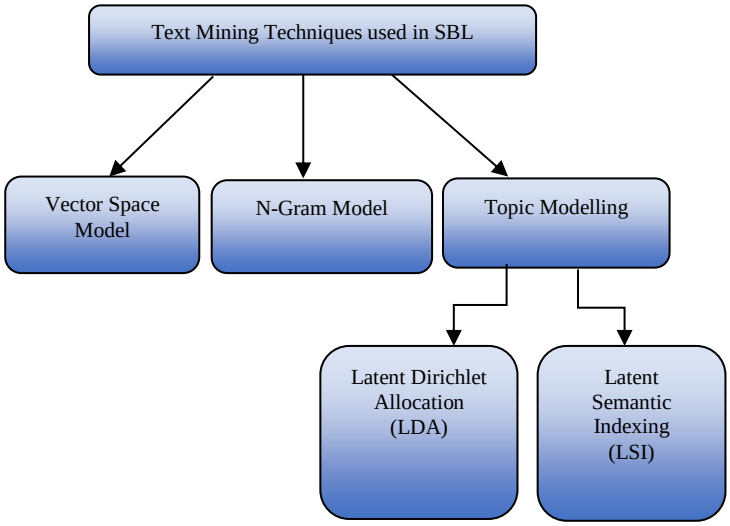

Fig. 1. Text Mining Techniques for SBL

probabilistic model and able to predict hidden topics while LSI is limited to generative modelling only. Another model proposed named bug Scout [2] which utilizes the concept of similarity (in terms of technical aspects explored as topics) between bug report and source code and Narrowing the ranking of source code files with the help of extended LDA. Old bug fixing information also proved to be a rich source of information and utilized for locating of new bugs. rVSM [3] based on classic VSM utilizes old bug information in association with document length and verified its results with four open source java projects. Addition of source code structured information [6] improved the accuracy of SBLM significantly. N-Gram models was very well explored for SBL[15] in which metadata ,title and description was taken

from bug reports, file content path and file name from source code. Similarity was computed by converting attributes in to character N-Gram conversion. One of the other way of utilizing the structured information is call graphs. LSI in conjunction with call graphs named LSICG [5] shows improved results on Rhino datasets in comparison with LSI. Sometimes one of the major problem is regional language of every region and especially for non-English regions. Indifference in the language of bug report and source code creates in big gap in localization of bugs. Cross Locator [8] used Microsoft and Google translators and then utilizes the combined results of both translators (in tf-idf form) for ranking of source code files. Even after using automated SBLM some of the actual buggy files ranked low in the list which resulted in wastage of resources like developer's time etc. and resulted in increase of software maintenance cost. To evaluate the effectiveness of SBLM APRILE [9] oracle was proposed and verified with 3000 bug reports of different java projects. Most of SBLM assume every source code file as a single unit irrespective of its size while large files are hard to localize because only small part of code contain the bug. This problem was solved by taking source code as segments [10] and proposed a new tool called BRTracer. It also consider stack trace information which is present in both BR and SC, proved to a valuable asset in increasing the similarity score. From all these experiments it was clear that hidden information like stack trace, structure information etc. played a significant role in increasing the accuracy of SBLM's. AmaLgam [11] best exemplifies above statement by combining all the information in an integrated form shows improved mean average precision of 16.4% than BLUiR. One of the hurdle or degrading performance factor of IR models was lexical mismatch or sematic gap of natural language and technical terms used between bug repositories and source code. To fill this gap there was a need of artificial intelligence or machine learning models like Learning to Rank (LTR) [12]. HyLoc [13] used rVSM for feature extraction and DNN to fill the semantic gap. Empirical Evaluation on six datasets named (AspectJ, Birt, Eclipse UI, JDT,SWT, Tomcat) shows improved results in terms of MRR and MAP in comparison with rVSM. Another integrated model named AmaLgam+ [16] utilizes five kind of information version history, similar report, source code structure information, stack trace and metadata. Localize bugs with the help of software changes shows promising results and evaluated on six open source projects in comparison with BRTracer, BLUiR, Amlagam. BLIA [22] utilizes texts, stack traces, comments in bug reports and structured information of source code files in addition of code change history.

Machine translation based SBLM [23] was proposed and recurrent neural network played as a heart of this model for localization of bugs.

III. FINDINGS

Proposed software bug localization models utilized text mining techniques in conjunction with other attributes. With the help of Table 1 we summarizes and extracted essential attributes employed in previous software bug localization models. Some of the major findings are described below.

TM Technique: Most widely used text mining technique as base model in previous SBLM's was Vector Space Model(as shown in fig.2) Although variations was made in terms of similarity score matrix etc. and most popular and efficient one was revised vector space model (rVSM) which was an extended version of classic VSM. Advanced TM techniques like LDA and LSI was also used and evaluated on benchmark datasets but VSM was used by majority of Researchers due to least complexity and efficient results. N-Gram models was not completely explored and there is still space for other text mining techniques to be used and evaluated for better results.

SR Employed: Software Repositories are the artifacts produced during software development life cycle. Source code and Bug report are main ingredients for every information retrieval based SBLM. After number of experiments on benchmark datasets it was evaluated and validated that other information like old bug fixing history, stack traces, structured information, change commit and version history etc. also played a vital role in increasing the accuracy of SBLM's.

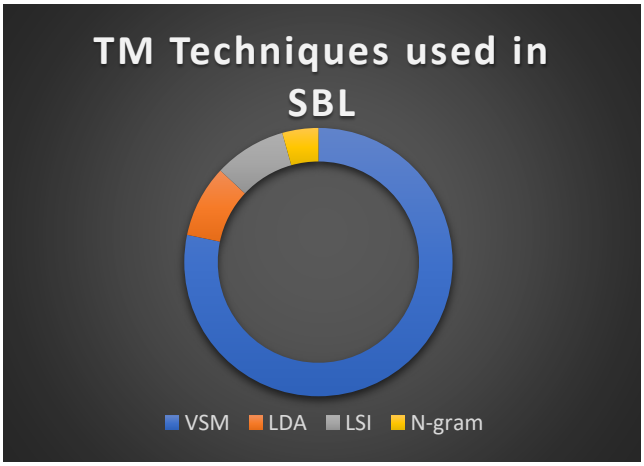

Fig. 2. Distribution of TM techniques used for SBL

Additional Techniques : Most widely used techniques used with text mining in SBLM's was Machine learning one of the best example was learning to rank model which give us direction to explore combination of both the techniques. In last two to three years deep learning with NLP based SBLM's able to filled up the gap of lexical mismatch between BR and SC. Although research is going on and lot of work is needed to be done in future in this direction. Because till now only small number of techniques are experimented (SVM, Naïve Bayes, CNN and DNN).

Datasets Used: Because of availability of Public datasets every approach was very well evaluated on at least three projects but most of them are java based. So, it questions about robustness of SBLM's on other languages. This is needed in future to do experiment with other languages also.

Efficacy Measures: Most widely used efficacy measure was Mean Reciprocal Rank (MRR), Mean Average Precision (MAP) and Rank @ k matrix in which K is number of buggy source files ranked corresponding to any bug report. Models which are based on ML + TM techniques used precision, recall and F-measure also.

Granularity level (G.L): Granularity level means level at which ranking is accomplished for buggy source code. Smaller the granularity better the localization because less effort is needed for backtracking of methods or classes as comparison with files. But mostly SBLM used files as raking

TABLE 1. STATE OF THE ART MODELS FOR SBL

SBLM	Text Mining Technique	Additional Information/Technique	Repositories Used	Data Set Employed	Efficacy Measures	Outperform By	G.L	S.M
Bug Scout [2]	Extended LDA	Defect Proneness	BR,SC	D1,D3,D7,D8	E1,E4	SVM and LDA+VSM	F.L	N
Bug Locator [3]	VSM	Document Length	BRe, BR,SC	D1,D3,D5,D12	E1,E2,E3	VSM,LDA,LSA	F.L	N
LIBCROOS[4]	LSI+VSM	Binary Class Relationship	SC, BR, BRe, Binary Code	D14,D15,D16, D17	E8	--	F.L	N
LSICG[5]	LSI	Call Graphs	SC, metadata, BR	D16	E1	LSI	F.L	Y
BLUiR[6]	VSM	Source code str. + AST	SC, metadata, BR	D1,D3,D5,D12	E1,E2,E3	--	F.L	N
Bug Localizer[7]	VSM	SS+ TS+ Metadata	BRe, BR, SC	D1,D3,D5	E5,E6,E7	--	F.L	N
Cross Locator[8]	VSM	SVM	BRe, BR, SC	D13	E2, E3	--	F.L	N
APRILE[9]	LDA	SVM+SS	BRe, BR, SC	D1,D3,D11	E5,E6,E7	BLUiR, AmaLgam	F.L	N
BRTracer[10]	VSM	Bug Locator +Segment	BR, SC, VHD, BRe, Metada	D1,D3,D5	E1,E2,E3	Bug Locator	F.L	N
AmaLgam[11]	VSM	VH+ SR+ Str. Information	BR, SC, VHD, BRe, Metada	D1,D3,D11,D12	E1,E2,E3	BLUiR	F.L	N
LTR[12]	VSM	SVM + API Entries	BR ,SC, BRe	D1,D2,D3,D4 D5,D6	E2,E3,E4	Bug Scout, Bug Locator	F.L	Y
HyLoc[13]	VSM	DNN+PI	BR,SC,VHD,BRe ,Metada	D1,D2,D3,D4 D5,D6	E1,E2,E3	DNN, rVSM	F.L	N
AML[14]	VSM	Spectra Information	BR, SC, VHD, BRe, Metada	D1,D15,D16	E1,E2	VSM	F.L	N
NP-CNN[15]	N-gram Model	CNN+ Str. Information	BR,HCS,SC	D1,D4,D9,D10	E1,E2,E4	Bug Locator CNN, HyLoc	F.L	Y
AmaLgam+[16]	VSM	GA+SS	BR,VHD,SC,ST	D1,D3,D11,D12	E1,E2,E3	AmaLgam, BLUiR, VSM	F.L	Y
Drew BL, Comb BL[17]	VSM+NLP	Bug Fixing History	CCH,SC,VHD,BR, Bre	D2,D6	E1,E2,E3	DNN, HyLoc	F.L	Y
Locus[18]	VSM	Software Changes	BR, SC, VHD, BRe, Metada	D1,D4,D5,D6, D10	E1,E2,E3	AmaLgam, BLUiR	F.L	N
Bug Catcher[19]	VSM	Indexing Approach	BR, SC ,Metadata	D1,D3,D5	E1,E2,E3	BLUiR, Bug Locator	F.L	N
DNNLOC[20]	VSM	DNN+TS	BR, SC ,Metadata	D1,D2,D3,D4, D5,D6	E1,E2,E3	BLUiR	F.L	N
Deep Locator[21]	VSM	CNN+AST	BR,SF	D1,D2,D4,D5, D6	E5,E6,E7	CNN, HyLoc	F.L	N
BLIA[22]	VSM	Metadata , Str. Info.	BR, BRe, SC,VHD	D1,D2,D3	E1,E2,E3	Bug Locator, BLUiR	F.L	Y
Bug Translator[23]	MT	RNN+API	BRe, BR, SC, Metadata	D3,D4,D5	E1,E2,E3	LTR, Word Embedding	F + M	N
CNN Forest[24]	VSM	CNN+ ERF+ Str. Info.	BR,SF, Metadata	D1,D2,D4,D5, D6	E2, E3	NP CNN,DNN LOC	FL	N

D1:AspectJ,D2:Birt,D3:Eclipse,D4:JDT,D5:SWT,D6:Tomcat,D7:ArgoUML,D8:Jazz,D9:PF,D10:PDE,D11:SWT,D12:ZXing,D13:RubyChina,D14:Jabref,D15:Lucene,D16:Rhino,D17:muCommander
E1:TOPK,E2:Mean Average Precision (MAP),E3:Mean Reciprocal Rank(MRR),E4:Accuracy,E5:Precision,E6:Recall,E7:F-measureE8:Standard Deviation, BR: Bug Report, SC: Source Code, HCS: History Control System, CCH :Code Change History, ST: Stack Trace, VHD: Version History Data, SS: Suspiciousness Score, SR: Similar Report, AST: Abstract Syntax Tree, CNN: Convolutional Neural Network ,RNN: Recurrent Neural Network, VSM: Vector Space Model, SVM: Support Vector Model, ERF: Ensemble of Random Forest, SBLM: Software Bug Localization Model, GL: Granularity Level, SM: Statistical Measure

granularity therefore work is needed in this direction to increase the granularity of buggy source code**.**

*Statistical Measure (S.M)***:** Until and unless a technique is not verified with some standard statistical test it doesn't give surety about its performance on the basis of datasets. Because datasets vary in nature according to projects and proportion of bugs with respect to source code changes. According to our study only thirty to forty percent SBLM's was validated with statistical measures**.**

IV. CONCLUSION

In this paper we study different software Bug Localization Models which uses text mining techniques in conjunction with software repositories and other Machine Learning Techniques. With the help of table 1 we extract and explain all the essential features (like text mining technique, software repositories used for mining, datasets employed, efficacy measures employed, granularity level of ranking, statistical measure used and additional technique/attribute) employed in previous software bug localization models. As we describe in previous section work is needed to be done in directions like Robustness of SBLM's, exploitation of ML + TM techniques, efficient use of other hidden information present in software repositories and on granularity level of source code. In future we will try to build a novel Bug localization model by taking in to consideration of these issues.

ACKNOWLEDGMENT

This work is sponsored by National Project Implementation Unit (NPIU) under TEQIP-III.

REFERENCES

[1] Lukins, Stacy K., Nicholas A. Kraft, and Letha H. Etzkorn. "Bug localization using latent dirichlet allocation." Information and Software Technology, 2010, pp. 972-990.

[2] Nguyen, Anh Tuan, Tung Thanh Nguyen, Jafar Al-Kofahi, Hung Viet Nguyen, and Tien N. Nguyen. "A topic-based approach for narrowing the search space of buggy files from a bug report." In Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering,. IEEE Computer Society, 2011, pp. 263-272.

[3] Zhou, Jian, Hongyu Zhang, and David Lo. "Where should the bugs be fixed? more accurate information retrieval-based bug localization based on bug reports." In Software Engineering (ICSE), 2012 34th International Conference on, IEEE, 2012, pp. 14-24.

[4] Ali, Nasir, Aminata Sabane, Yann-Gael Gueheneuc, and Giuliano Antoniol. "Improving bug location using binary class relationships." In Source Code Analysis and Manipulation (SCAM), 2012 IEEE 12th International Working Conference on, IEEE, 2012, pp. 174-183.

[5] Shao, Peng, Travis Atkison, Nicholas A. Kraft, and Randy K. Smith. "Combining lexical and structural information for static bug localisation." International Journal of Computer Applications in Technology 44, 2012, pp. 61-71.

[6] Saha, Ripon K., Matthew Lease, Sarfraz Khurshid, and Dewayne E. Perry. "Improving bug localization using structured information retrieval." In Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on, IEEE, 2013, pp. 345-355.

[7] Thung, Ferdian, Tien-Duy B. Le, Pavneet Singh Kochhar, and David Lo. "BugLocalizer: integrated tool support for bug localization." In Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, ACM, 2014, pp. 767-770.

[8] Xia, Xin, David Lo, Xingen Wang, Chenyi Zhang, and Xinyu Wang. "Cross-language bug localization." In Proceedings of the 22nd International Conference on Program Comprehension, ACM, 2014, pp. 275-278.

[9] Le, Tien-Duy B., Ferdian Thung, and David Lo. "Predicting effectiveness of ir-based bug localization techniques." In 2014 IEEE 25th International Symposium on Software Reliability Engineering (ISSRE), IEEE, 2014, pp. 335-345.

[10] Wong, Chu-Pan, Yingfei Xiong, Hongyu Zhang, Dan Hao, Lu Zhang, and Hong Mei. "Boosting bug-report-oriented fault localization with segmentation and stack-trace analysis." In 2014 IEEE International Conference on Software Maintenance and Evolution (ICSME), IEEE, 2014, pp. 181-190.

[11] Wang, Shaowei, and David Lo. "Version history, similar report, and structure: Putting them together for improved bug localization." In Proceedings of the 22nd International Conference on Program Comprehension, ACM, 2014, pp. 53-63.

[12] Ye, Xin, Razvan Bunescu, and Chang Liu. "Learning to rank relevant files for bug reports using domain knowledge." In Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, ACM, 2014, pp. 689-699.

[13] Lam, An Ngoc, Anh Tuan Nguyen, Hoan Anh Nguyen, and Tien N. Nguyen. "Combining deep learning with information retrieval to localize buggy files for bug reports (n)." In Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on,. IEEE, 2015, pp. 476-481.

[14] Le, Tien-Duy B., Richard J. Oentaryo, and David Lo. "Information retrieval and spectrum based bug localization: better together." In Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ACM, 2015, pp. 579-590.

[15] Huo, Xuan, Ming Li, and Zhi-Hua Zhou. "Learning Unified Features from Natural and Programming Languages for Locating Buggy Source Code." In IJCAI, 2016, pp. 1606-1612.

[16] Wang, Shaowei, and David Lo. "AmaLgam+: Composing rich information sources for accurate bug localization." Journal of Software: Evolution and Process 28, 2016, pp. 921-942.

[17] Uneno, Yukiya, Osamu Mizuno, and Eun-Hye Choi. "Using a distributed representation of words in localizing relevant files for bug reports." In Software Quality, Reliability and Security (QRS), 2016 IEEE International Conference on, IEEE, 2016, pp. 183-190.

[18] Wen, Ming, Rongxin Wu, and Shing-Chi Cheung. "Locus: Locating bugs from software changes." In Automated Software Engineering (ASE), 2016 31st IEEE/ACM International Conference on, IEEE, 2016, pp. 262-273.

[19] Kılınç, Deniz, Fatih Yücalar, Emin Borandağ, and Ersin Aslan. "Multi - level reranking approach for bug localization." Expert Systems 33, 2016, pp. 286-294.

[20] Lam, An Ngoc, Anh Tuan Nguyen, Hoan Anh Nguyen, and Tien N. Nguyen. "Bug localization with combination of deep learning and information retrieval." In Program Comprehension (ICPC), 2017 IEEE/ACM 25th International Conference on, IEEE, 2017, pp. 218-229.

[21] Xiao, Yan, Jacky Keung, Qing Mi, and Kwabena E. Bennin. "Improving Bug Localization with an Enhanced Convolutional Neural Network." In Asia-Pacific Software Engineering Conference (APSEC), 2017 24th, IEEE, 2017, pp. 338-347.

[22] Youm, Klaus Changsun, June Ahn, and Eunseok Lee. "Improved bug localization based on code change histories and bug reports." Information and Software Technology 82, 2017, pp.177-192.

[23] Xiao, Yan, Jacky Keung, Kwabena E. Bennin, and Qing Mi. "Machine translation-based bug localization technique for bridging lexical gap." Information and Software Technology 99, 2018, pp.58-61.

[24] Xiao, Yan, Jacky Keung, Qing Mi, and Kwabena E. Bennin. "Bug Localization with Semantic and Structural Features using Convolutional Neural Network and Cascade Forest." In Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018, ACM, 2018, pp. 101-111.

[25] Hoang, Thong Van-Duc, Richard J. Oentaryo, Tien-Duy Bui Le, and David Lo. "Network-Clustered Multi-Modal Bug Localization." IEEE Transactions on Software Engineering, 2018.