

Empirical Study And Analysis Of Software Bug Localization Approaches Using Deep Learning

^[1]Ginika Mahajan, ^[2]Neha Chaudhary, ^[3]Anita Shrotriya

^{[1][2][3]} Manipal University Jaipur, India

E-mail: ^[1]ginika.5aug@gmail.com

Abstract: Bugs comes in high volume and documenting them properly in a specific format in the bug reports is difficult. Locating these bugs in correctly identified buggy files is a challenging task which needs to be automated. Numerous tools and techniques are proposed by researchers to support developers and testers to detect buggy files and automate the process of bug localization with greater accuracy. Recent research deal with the automation of Bug Localization process by using different techniques and tools. In this paper, we presented a comprehensive review of few papers in the domain of bug localization. This review helps us to know the benchmark datasets that are used in this process of bug localization, the major techniques that are worked upon, the findings and evaluation criteria and the architecture of various models and frameworks developed by researchers to automate the task of bug localization. This paper works on IR and DNN approaches and attempted to improve the previous results of DNNLOC successfully. The optimized version of the model improved the accuracy from 0.815 to 0.969 for enhanced rvsm and 0.83 to 0.971 for enhanced dnn. It is apparent that information retrieval approach and deep learning approach complement together in the domain of bug localization.

Keywords: Bug Localization, enhanced rVSM, IR DNN.

1. Introduction

The activity of finding and locating bugs correctly is a difficult and time-exhausting task. Great deal of resources and time is wasted in managing this activity, especially if it is done manually and without using any tool. Bugs comes in high volume and documenting them properly in a specific format in the bug reports is difficult. Locating these bugs in correctly identified buggy files is a challenging task which needs to be automated. Numerous tools and techniques are proposed by researchers to support developers and testers to detect buggy files. These bug localization tools and techniques evaluates the symptoms of a bug and generates ranked list of files on the basis of probability of occurrence of bug.

For large and real time projects, developers need to explore and examine a huge amount of source files, find the buggy files from the symptoms given in bug reports. Automated tools are required to ease the manual process of finding and understanding the source of bug. There are various instances where users realize that the data received by them faces glitches due to the minor faults in the source code or files. This is where the bug problem initiates, which made us research about the issue in depth to see how bug problems are solved in the corporate world where the developer faces a lot of challenges to localize a bug in hundreds of files.

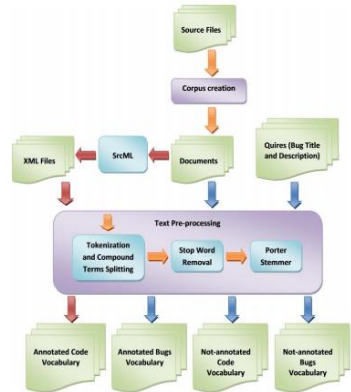
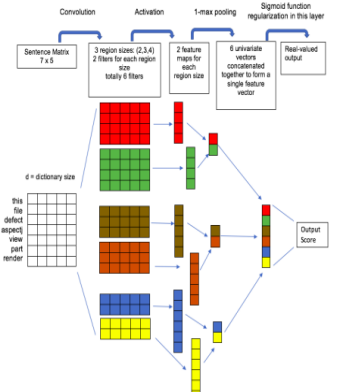
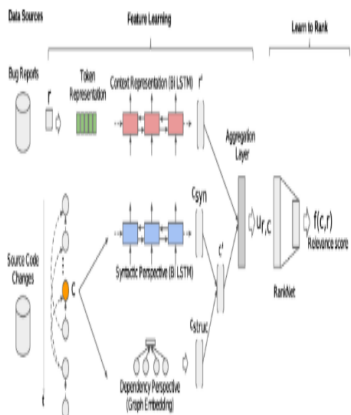
Several approaches in the analyses and experiments to automate bug localization are proposed. Presently, advanced information retrieval techniques [4][7][8][11][12], machine learning techniques [6][11], deep learning techniques [2][3][5][13][14] have been used to ascertain relation of bug in source code files.

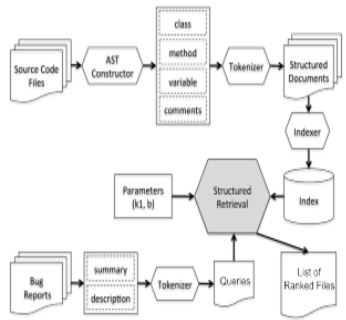
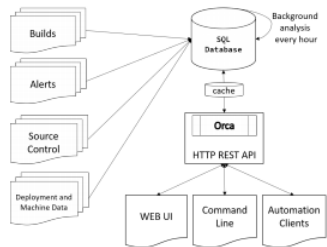
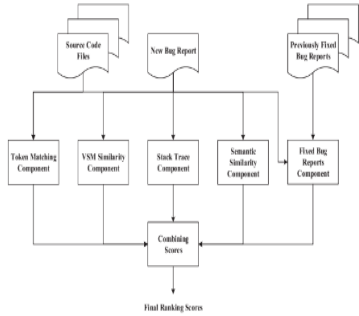
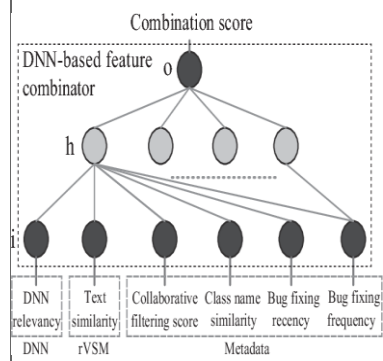
2. Related Work

We conducted a detailed survey on various research papers on Bug Localization. Table 1 demonstrates the detailed survey of each paper based on datasets used, techniques implemented, results in terms of accuracy, TopK, etc and architecture used by different authors. Also, we have gone through the existing techniques and tried to improve and enhance them. This paper also presents the implementation and enhancement of DNNLoC with improved results of automating bug localization process.

Table 1: Detailed survey of research papers on Bug Localization

Paper	Dataset	Tool & Technique	Results	Architecture
Network-clustered multi-modal bug localization [1]	Dataset includes <ul style="list-style-type: none"> Ant AspectJ Lang Lucene Math Rhino Time. 	Joint optimization of BL error and cluster of bug reports and program components. network Lasso Regularization is applied for clustering.	The results show that NetML exceeds the best-performing baseline by 31.82%, 22.35%, 19.72%, 19.24%, in terms of the number of bugs effectively localized while inspecting the top 1, 5, and 10 methods and Mean Average Precision (MAP)	
Improving Bug Localization with an Enhanced Convolutional Neural Network [2]	Dataset includes <ul style="list-style-type: none"> AspectJ Eclipse JDT SWT Tomcat 	Enhanced CNN with bug-fixing experience, new rTF-IDuF method and pretrained word2vec technique.	DeepLocator attains 9.77% to 26.65% greater measure than the conventional CNN and 3.8% higher MAP than a state-of-the-art method HyLoc use up less computation time.	
Deep Learning with Customized Abstract Syntax Tree for Bug Localization [3]	Dataset includes <ul style="list-style-type: none"> AspectJ SWT JDT Tomcat 	combines tree based CNN with customized ASTs	CAST attained higher MAP of 0.044 MRR of 0.033 than most excellent results of 4 existing state-of-art techniques (BugLocator, DNNLOC, DeepLocator and NP-CNN).	

On the Relationship Between the Vocabulary of Bug Reports and Source Code [4]	<p>Dataset includes</p> <ul style="list-style-type: none"> • ADempiere 3.10 • Art of Illusion 2.4.1 • JEdit 4.2 • aTunes 1.10 • Eclipse 2.0 • Eclipse 3.5 	Text retrieval technique	The report confirms that source code share various terms along with bug reports, the number of shared terms is greater for patched classes than for rest of classes.	
On Usefulness of the Deep-Learning-Based Bug Localization Models to Practitioners [5]	<p>Dataset includes</p> <ul style="list-style-type: none"> • AspectJ • Tomcat • SWT • Eclipse • JDT. 	Convolution Neural Network and Simple Logistic model to analyze their efficiency.	Though deep learning models perform well than simple machine learning models, they meet the acceptance criteria placed by the practitioners partly.	
Bug Localization by Learning to Rank and Represent Bug Inducing Changes [6]	<p>Dataset A includes</p> <ul style="list-style-type: none"> • Zxing • SWT • AspectJ • PDE • JDT • Tomcat <p>Dataset B includes</p> <ul style="list-style-type: none"> • AspectJ • Birt • JDT 	IR and CNN based technique is used.	<p>Dataset A, the end-to-end configuration goes up to a 21% improved MAP for the SWT project, while on average the differences are 12% and 15% in for MAP and MRR</p> <p>Dataset B, results are consistent, also showing an average increase of more than 10% for both metrics.</p>	

Improving bug localization using structured information retrieval [7]	Dataset Includes <ul style="list-style-type: none"> • SWT • Eclipse • AspectJ • Zxing 	IR based on code constructs, such as class and method names.	BLUiR matches or outperforms a current state-of-the-art tool across applications considered, even when BLUiR does not use bug similarity data used by the other tool.	
Orca: Differential Bug Localization in Large-Scale Services [9]	NA	Uses differential code analysis and build provenance graph	Appropriately localizes 77% of bugs Causes 4x reduction in the work done by the OCE.	
Leveraging textual properties of bug reports to localize relevant source files [12]	Dataset Includes <ul style="list-style-type: none"> • AspectJ • SWT • Zxing 	IR, textual matching, stack trace analysis, and multi-label classification	Rank appropriate source files for more than 52% of bugs by suggesting only one source file and 78% by recommending ten files. Also improves MRR and MAP values.	
Bug Localization with Combination of Deep Learning and Information Retrieval [14]	Dataset Includes <ul style="list-style-type: none"> • AspectJ • Birt • Eclipse UI • JDT • SWT • Tomcat. 	DNNwith rVSM	DNNLOC achieves higher accuracy than the state-of-the-art IR and machine learning techniques	

3. Model Descriptions

Recent studies deal with the automation of Bug Localization process by using different techniques and tools. In [4][7][8][10][12], authors proposed advanced information retrieval technique to automate the process of Bug Localization. The results of studies implementing IR showcase that the power of this approach in finding the lexical similarity between bug reports and source code. Also, studies shows deep neural networks paired with IR approach gives good results on bug localization. The study we researched and implemented attains different features by using IR and other text processing approaches and then by using DNN on same dataset, the results are improved.

In the study [14] of Lam et al., authors discussed bug localization model DNNLOC that uses dnn and rvsm to handle the lexical mismatch and text to similarity in the bug reports. We studied several recent research in the domain of bug localization and focused on the paper 'Bug localisation with a combination of deep learning And Information retrieval' for improvement. We find few research gaps and scope of improvement especially in terms of accuracy of the given model. We studied the impact and need of various parameters and techniques on accuracy used in the model of DNNLOC. We have optimized the model by enhancing the techniques used and attained accuracy from 0.808 to 0.971. We implemented with some enhancements rvSM and this model which results in increase in accuracy of the bug localization techniques.

4. Dataset

The dataset incorporates bug reports of six distinct project of AspectJ, Birt, Eclipse Platform UI, JDT, SWT and Tomcat. We used same dataset as used in the paper for comparisons with results of our implementation and the original work. Fig.1 shows a section of rows from the used dataset

id	bug_id	summary	description	report_time	report_time	status	commit	commit_time	files
1	422205	Bug 422205 Missing help content reference in N	The help context ID is not referenced by the Mc	2013-11-20 20:05:32	1385000000	resolved fixed	345f01b	1385000000	data/org.eclips
2	389229	Bug 389229 Documentation of possible values/objects of property IDeviceRenderer.FILE_IDENTI		2012-09-11 05:16:02	1347360000	resolved fixed	37c1c08	1382650000	chart/org.eclips
3	403306	Bug 403306 Nested DataEngineSessions and wr	(I have following use case: - Data Cube with mea	2013-03-14 05:52:07	1363250000	closed fixed	8299b4a	1381480000	data/org.eclips
4	375294	Bug 375294 JBoss + Sybase database issue	Build Identifier: 3.7.100.v20110510-0712 Hi, JBo	2012-03-26 05:12:15	1332750000	resolved fixed	ffe1c1f	1367050000	data/org.eclips
5	416987	Bug 416987 ArrayIndexOutOfBoundsException	There is a bug in org.eclipse.birt.report.engine.i	2013-09-11 04:59:50	1378890000	resolved fixed	5eeefcf	1380240000	engine/org.eclips
6	416703	Bug 416703 MongoDB : Display correctly the Number data type		2013-09-06 05:27:54	1378460000	resolved fixed	438114f	1379550000	data/org.eclips
7	417317	Bug 417317 Error when running chart example SwingInteractivityViewer.java.		2013-09-16 06:05:24	1379330000	verified fixed	ec88e80	1379340000	chart/org.eclips

Fig 1: Section of bug report of dataset

5. Experiment

In Bug Localization process, a bug report is paired with each source file. It extracts the features from the report and uses information retrieval techniques for conversion of feature vectors for each pair. Score for each file with respect to the bug report is then calculated. The scores for all the source files are ranked and the high-ranking score indicates that particular file is potentially buggy for the bug report.

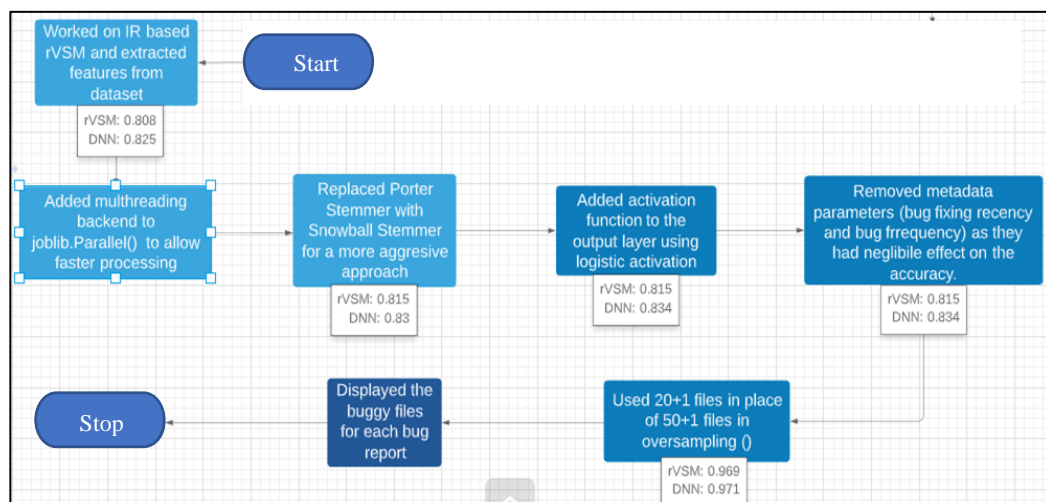


Fig 2: Workflow Process

To extract the textual features from Bug Reports, the model parses the report to remove whitespaces, stopwords, punctuation marks, etc. The workflow process is shown in the fig.2 to allow faster processing. Multithreading backend to function `joblib.parallel()` is added. Instead of using Porter stemming algorithm to convert words to their base stem, we chose to implement Snowball Stemmer which has more aggressive approach. By replacing porter stemmer with snowball stemmer, we optimized minor increase in accuracy results of rvsm from 0.808 to 0.815 and dnn from 0.825 to 0.830. Then the model extracts code and textual features from source files.

rVSM is revised Vector Space Model, an Information Retrieval approach based on cosine similarity which is shown in Equation (1)

$$Similarity = \cos(q, d) = \frac{V_q * V_d}{(\|V_q\|) * (\|V_d\|)} \quad (1)$$

Equation (2) is used to compute length value of every source file as per number of terms in it.

$$g(\#terms) = \frac{1}{1 + e^{-N.(\#terms)}} \quad (2)$$

Combining the above two equations, rVSM score is calculated as given in equation (3).

$$rVSMscore(q, d) = g(\#terms) \times \cos(q, d) \quad (3)$$

Textual similarity in the bug report and corresponding source file is considered as a feature in this model. As rVSM gives better results than VSM, the textual similarity of a bug report and a file is calculated by using rVSM. By using this, the score of the pair (bug report, file) is calculated. Added activation function to the output layer using logistic activation. The model considers various metadata parameters such as bug fixing recency, bug frequency, collaborative filtering, and class name similarity. We find that two of these metadata parameters were having negligible effect in the process and are adding load to the process. By removing this metadata that has negligible effect in the process the accuracy is increased from 0.830 to 0.834. We identified the problem of oversampling in the model which has a significant effect on the accuracy. We used 20+1 files in place of 50+1 files in oversampling().

6. Results

For evaluation in bug localization processes, Top-k accuracy score is considered as the main evaluation criteria. This metric calculates the correct occurrence of label among top k labels as ranked by predicted scores. Consequently, we computed and evaluated accuracies in terms of Top-k score. We implemented the two models, DNN Based Model and rVSM Based Model and applied modification for improvements in accuracy results. The top-k accuracy of the enhanced rvsm and enhanced dnn model is shown in the fig.3.

```
rVSM Model:
{1: 0.423, 2: 0.553, 3: 0.623, 4: 0.675, 5: 0.726, 6: 0.759, 7: 0.776, 8: 0.798, 9: 0.816, 10: 0.83, 11: 0.848, 12: 0.858, 13: 0.871, 14: 0.883, 15: 0.893, 16: 0.903, 17: 0.909, 18: 0.917, 19: 0.919, 20: 0.924, 21: 0.969}
DNN Model:
{1: 0.486, 2: 0.611, 3: 0.677, 4: 0.719, 5: 0.758, 6: 0.785, 7: 0.802, 8: 0.822, 9: 0.836, 10: 0.848, 11: 0.864, 12: 0.872, 13: 0.885, 14: 0.896, 15: 0.904, 16: 0.912, 17: 0.917, 18: 0.923, 19: 0.925, 20: 0.929, 21: 0.971}
```

Fig 3: Top-k results of enhanced model (K=1 to 21)

The line chart graph plotted between accuracy and k-value is shown in fig.4. The plot shows increase in accuracy of enhanced rvsm and enhanced dnn. The accuracy was improved from 0.815 to 0.969 for rvsm and 0.83 to 0.971 for dnn.

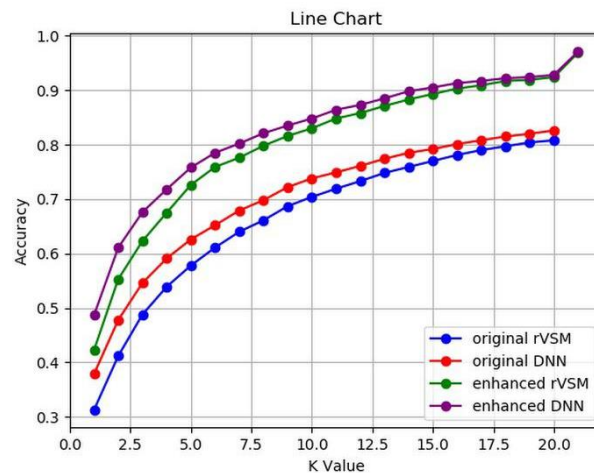


Fig 4: Comparison of accuracy of original and enhanced model

7. Conclusion

In this paper, we presented a comprehensive review of few papers in the domain of bug localization. This review helps us to know the benchmark datasets that are used in this process of bug localization, the major techniques that are worked upon, the findings and evaluation criteria and the architecture of various models and frameworks developed by researchers to automate the task of bug localization.

In this study, we worked on IR techniques and DNN and attempted to improve the results of previous paper successfully. The accuracy was improved from 0.815 to 0.969 for enhanced rvsm and 0.83 to 0.971 for enhanced dnn. Hence it is evident that information retrieval approach and deep learning approach complement together in the domain of bug localization.

References

- [1] Hoang, T., Oentaryo, R. J., Le, T. D. B., & Lo, D. (2018). Network-clustered multi-modal bug localization. *IEEE Transactions on Software Engineering*, 45(10), 1002-1023.
- [2] Xiao, Y., Keung, J., Mi, Q., & Bennin, K. E. (2017, December). Improving bug localization with an enhanced convolutional neural network. In *2017 24th Asia-Pacific Software Engineering Conference (APSEC)* (pp. 338-347). IEEE.
- [3] Liang, H., Sun, L., Wang, M., & Yang, Y. (2019). Deep learning with customized abstract syntax tree for bug localization. *IEEE Access*, 7, 116309-116320.
- [4] Moreno, L., Bandara, W., Haiduc, S., & Marcus, A. (2013, September). On the relationship between the vocabulary of bug reports and source code. In *2013 IEEE International Conference on Software Maintenance* (pp. 452-455). IEEE.
- [5] Polisetty, S., Miranskyy, A., & Başar, A. (2019, September). On usefulness of the deep-learning-based bug localization models to practitioners. In *Proceedings of the Fifteenth International Conference on Predictive Models and Data Analytics in Software Engineering* (pp. 16-25).
- [6] Loyola, P., Gajananan, K., & Satoh, F. (2018, October). Bug localization by learning to rank and represent bug inducing changes. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (pp. 657-665).
- [7] Saha, R. K., Lease, M., Khurshid, S., & Perry, D. E. (2013, November). Improving bug localization using structured information retrieval. In *2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)* (pp. 345-355). IEEE.

- [8] Dao, T., Zhang, L., & Meng, N. (2017, May). How does execution information help with information-retrieval based bug localization?. In *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)* (pp. 241-250). IEEE.
- [9] Bhagwan, R., Kumar, R., Maddila, C. S., & Philip, A. A. (2018). Orca: Differential bug localization in large-scale services. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)* (pp. 493-509).
- [10] Wen, M., Wu, R., & Cheung, S. C. (2016, September). Locus: Locating bugs from software changes. In *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)* (pp. 262-273). IEEE.
- [11] Singh, A., Bhatia, R., & Singhrova, A. (2018). Taxonomy of machine learning algorithms in software fault prediction using object oriented metrics. *Procedia computer science*, 132, 993-1001.
- [12] Gharibi, R., Rasekh, A. H., Sadreddini, M. H., & Fakhrahmad, S. M. (2018). Leveraging textual properties of bug reports to localize relevant source files. *Information Processing & Management*, 54(6), 1058-1076.
- [13] Xiao, Y., Keung, J., Bennin, K. E., & Mi, Q. (2018). Machine translation-based bug localization technique for bridging lexical gap. *Information and Software Technology*, 99, 58-61.
- [14] Lam, A. N., Nguyen, A. T., Nguyen, H. A., & Nguyen, T. N. (2017, May). Bug localization with combination of deep learning and information retrieval. In *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)* (pp. 218-229). IEEE.