Poster: Are Information Retrieval-based Bug Localization Techniques Trustworthy?

Misoo Kim
Sungkyunkwan University
Suwon, South Korea
misoo12@skku.edu

Eunseok Lee
Sungkyunkwan University
Suwon, South Korea
leees@skku.edu

ABSTRACT

Information retrieval-based bug localization techniques are evaluated using datasets with an oracle. However, datasets can contain non-buggy files, which affect the reliability of these techniques. To investigate the impact of non-buggy files, we show that a test file can be regarded as a buggy file. Then, we determined if this file caused inaccuracies that would eventually affect the trustworthiness of previous techniques. We further analyzed the impact of test files on IR-based bug localization through three research questions. Our results show that the test files significantly impact the performance of the techniques. Furthermore, MAP increased by a maximum of 21%, and MRR decreased by a maximum of 13%.

CCS CONCEPTS

• **Software and its engineering → Software testing and debugging**; • **General and reference** → *Empirical studies*;

KEYWORDS

Empirical Study, Test File, Bug Report, Information Retrieval-based Bug Localization, Trustworthness

1 INTRODUCTION

Automatically locating a buggy file reduces the cost of bug resolution. So, researchers have proposed information retrieval-based bug localization (IR-based BL) techniques [1–5]. To significantly improve the bug resolution process in a real software project, it is necessary to guarantee the trustworthiness of techniques [6].

The performance of IR-based BL techniques was evaluated using a dataset and was compared with other techniques that were evaluated using the same datasets. The datasets were designed with both an oracle and a collection of source files for a target project [7]. We examined the validity of popular datasets used for the evaluation of past studies and identified a case in which test files are regarded as buggy files and are contained in the collection.

Test files are run to ensure that there are no bugs before software is released [8], and they do not influence the use of released software. That is, the test file might not be suitable for locating production-buggy files. In this situation, developers might not trust IR-based production BL techniques.

ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5663-3/18/05.
https://doi.org/10.1145/3183440.3194954

Even though the trustworthiness of IR-based BL techniques can be called into question due to exist test files, no research has examined the impact of these issues. In this work, we analyzed the impact of test files on IR-based bug localization techniques. Our study addresses the following research questions:

RQ1 How much is the test files involved in the performance evaluation of IR-based bug localization techniques?
RQ2 Do test files affect the performance of IR-based bug localization techniques?
RQ3 Are IR-based BL techniques trustworthy?

Our contributions include: analyzing how many test files are included in the popular datasets; evaluating the performance of previous techniques depending on test files; and confirming performance of the techniques without test files.

2 BACKGROUND

2.1 IR based Bug Localization

An IR-based BL searches buggy files by utilizing a bug report as a query and a set of source files as the document collection and using diverse information as features to more accurately search buggy files [1–5]. BugLocator uses both the revised vector space model and a set of historical bug reports and a new bug report [1]. BLUiR locates buggy files using structured information based on both a code structure and a bug report structure[3]. BRTracer uses the stack trace[4]. BLIA+ integrated previously mentioned features, commit historys and comments on bug reports[5]. These techniques are compared using the same datasets and evaluation metrics to validate that they are better than previous techniques.

2.2 Performance Evaluation

Popular evaluation metrics for IR-based bug localization techniques are Top 10 rank rate (Top10), mean average precision (MAP), and mean reciprocal rank (MRR). Previous techniques used a dataset provided by Zhou et al.[1]. The dataset is based on four projects, such as Eclipse, AspectJ, SWT, and Zxing.

3 EXPERIMENTAL RESULTS

3.1 RQ1. Involvement of Test Files in Dataset

Table 1 summarizes the number of test files(TF) in a dataset that consist of oracles and a source file collection. In the dataset, Eclipse and SWT contained few test files in the collective buggy files and source files being searched. In the Zxing and AspectJ projects, 15% and 55% of bug reports contained test files as buggy files, respectively, and 25% and 13% of the source files were test files.

Table 1: Involvement of test files in Datasets

Project	# Bugs			# Files	
	Total	with TF	without TF	Total	TF
Eclipse	3,075	21	3,071	12,863	66
AspectJ	286	157	284	5,188	679
SWT	98	0	98	738	6
Zxing	20	3	19	391	96
Total	**3,479**	**181**	**3,471**	**18,617**	**847**

Table 2: Performance change with effect size (Cohen's *d*)

Techniques	AspectJ		Zxing		d	
	MAP	MRR	MAP	MRR	MAP	MRR
BugLocator	3%	-13%	4%	-2%	**0.52**	**0.44**
BLUiR	14%	-6%	9%	-	**0.58**	**0.54**
BRTracer	5%	-13%	4%	-	**0.51**	**0.44**
BLIA+	21%	-	1%	-	**0.46**	**0.40**
Total	**10.8%**	**-8%**	**4.3%**	**-0.5%**	-	-

We confirmed that the dataset contains more than 10% test files in the AspectJ and Zxing projects for both buggy files in the oracle and the source file repository. This result means that the test files are involved in the evaluation method. Therefore, the validity of current datasets is violated by the inclusion of test files.

3.2 RQ2. IR-based BL Performance Evaluation

After finding that the current datasets are invalid, we analyzed the influence of this dataset on the performance of bug localization to confirm the reliability of evaluation method. We performed an evaluation of two cases: : 1) the test files are included in the dataset, 2) the test files are excluded from the dataset.

Table 2 shows the evaluation results about performance changes based on whether the test file is removed from the dataset. Our result shows just two projects, because we observed that the performance greatly varies in AspectJ and Zxing but not in Eclipse and SWT. For cases where the test file is not considered a buggy file and a source file collection, MAP increased by about 1-21%. the MRR dropped by about 0-13% except BLIA+. Our statistical analysis showed significant differences in both MAP and MRR for all cases, and that the effect sizes was medium(>0.3) or large(>0.5). Therefore, if the test file is not taken into consideration as a buggy file, the performance varied compared with the original results.

3.3 RQ3. Trustworthiness of the IR-based BL

Analysis results in the previous section showed that previous techniques have employed invalid datasets and unreliable evaluation methods. To discuss the trustworthiness of previous techniques, we compared the performance of existing techniques. We confirmed that BLUiR showed lower performance than BugLocator and BRTracer, which contradics previous results. This means that the datasets with test files can affect the performance evaluation for production bugs. Therefore, this situation can undermine the trustworthiness of the IR-based BL.

4 CONCLUSION

Many studies in the past have proposed IR-based BL techniques to rapidly address bugs. We confirmed that the dataset used to evaluate the performance includes test files that could undermine the validity of a dataset and the resulting trustworthiness of the performance evaluation. This situation results in developers that do not trust this technique.

We confirmed that the test file is regarded as a buggy file, and the test file includes a collection of source files. Based on these results, we evaluated the performance of IR-based production bug localization techniques based on whether or not the test file was applied. As a result, MAP increased, and we confirmed that the time required to find all buggy files compared to the previously evaluated performance can be reduced. MRR decreased, and we confirmed that it takes more time to find at least one buggy file compared to the existing performance evaluation. Based on our analysis, we found that the datasets are invalid, resulting in unreliable evaluation methods. Finally, we confirmed that the existing techniques are not trustworthy by confirming that the performance ranking of the previously evaluated techniques using invalid datasets. We confirmed that the performance evaluation can be maintained by using a dataset without test files to evaluate the performance of the production bug localization in the future. Using this method, the reliability and trustworthiness can be guaranteed.

ACKNOWLEDGMENTS

This research was supported by National Program (2015-0-00914) for Excellence in SW supervised by the Institute for IITP funded by MIST, Next-Generation Information Computing Development Program (2017M3C4A7065980) through the NNRF of Korea funded by the Ministry of Science, ICT and Basic Science Research Program (2016R1D1A1B03934610) through the NRF funded by the Ministry of Education. We are extremely grateful to Zhou et al., Saha et al., Wong et al., and Youm et al. for access to their datasets and results.

REFERENCES

[1] J. Zhou, H. Zhang, and D. Lo. Where should the bugs be fixed? more accurate infor- mation retrieval-based bug localization based on bug reports. In *Proceedings of the 34th International Conference on Software Engineering*, ICSE'12, pages 14–24, Zurich, Switzerland, June 2012.

[2] B. Sisman and A. C. Kak. Incorporating version histories in information retrieval based bug localization. In *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories*, MSR'12, pages 50–59, Zurich, Switzerland, June 2012.

[3] R. K. Saha, M. Lease, S. Khurshid, and D. E. Perry. Improving bug localization using structured information retrieva. In *Proceedings of the 2013 IEEE/ACM 28th International Conference on Automated Software Engineering*, ASE'13, pages 345–355, Silicon Valley, CA, USA, Jan 2013.

[4] C.P. Wong, Y. Xiong, H. Zhang, D. Hao, L. Zhang, and H. Mei. Boosting bugreport oriented fault localization with segmentation and stacktrace analysis. In *Proceedings of the 2014 IEEE International Conference on Software Maintenance and Evolution*, ICSEME'14, pages 181–190, Victoria, BC, Canada, Dec 2014.

[5] K. C. Youm, J. Ahn, and E. Lee. Improved bug localization based on code change histories and bug reports. *Information and Software Technology*, 82:177–192, February 2017.

[6] P. S. Kochhar, X. Xia, D. Lo, and S. Li. Practitioners' expectations on automated fault localization. In *Proceedings of the 25th International Symposium on Software Testing and Analysis*, ISSTA'16, pages 165–176, New York, NY, USA, July 2016.

[7] V. Dallmeier and T. Zimmermann. Extraction of bug localization benchmarks from history. In *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*, ASE'07, pages 433–436, Atlanta, Georgia, USA, 2007.

[8] Z. Gao, Y. Zou Z. Chen, and A. M. Memon. Sitar: Gui test script repair. *IEEE Transactions on Software Engineering*, 42(2):170–186, August 2015.