Poster: Improving Bug Localization with Report Quality Dynamics and Query Reformulation

Mohammad Masudur Rahman
University of Saskatchewan, Canada
masud.rahman@usask.ca

Chanchal K. Roy
University of Saskatchewan, Canada
chanchal.roy@usask.ca

ABSTRACT

Recent findings from a user study suggest that IR-based bug localization techniques do not perform well if the bug report lacks rich structured information such as relevant program entity names. On the contrary, excessive structured information such as stack traces in the bug report might always not be helpful for the automated bug localization. In this paper, we conduct a large empirical study using 5,500 bug reports from eight subject systems and replicating three existing studies from the literature. Our findings (1) empirically demonstrate how quality dynamics of bug reports affect the performances of IR-based bug localization, and (2) suggest potential ways (e.g., query reformulations) to overcome such limitations.

CCS CONCEPTS

•**Software and its engineering → Software verification and validation; Software testing and debugging;** *Software defect analysis;* Software maintenance tools;

ACM Reference format:
Mohammad Masudur Rahman and Chanchal K. Roy. 2018. Poster: Improving Bug Localization with Report Quality Dynamics and Query Reformulation. In *Proceedings of 40th International Conference on Software Engineering Companion, Gothenburg, Sweden, May 27-June 3, 2018 (ICSE '18 Companion),* 2 pages.
DOI: 10.1145/3183440.3195003

1 INTRODUCTION

Information Retrieval (IR)-based bug localization techniques rely on shared terms between a bug report and the project source [3, 8]. They are cheap and their performances are reported to be as good as that of spectra-based techniques [5]. Unfortunately, Wang et al. [7] has recently reported two major limitations based on a qualitative study where they involved human developers. First, IR-based techniques cannot perform well without the presence of rich structured information (e.g., program entity names pointing to defects or failures) in the bug report. Second, they also might not perform well with a bug report that contains excessive structured information (e.g., stack traces). In this paper, we conduct a large empirical study, and re-investigate the above issues. In particular, we demonstrate how quality dynamics of bug reports (i.e., prevalence of structured information or lack thereof) influence the performances of three existing IR-based techniques for bug localization [3, 6, 8], and then also discuss potential solutions (e.g., query reformulations).

2 EMPIRICAL STUDY

Wang et al. [7] investigate IR-based bug localization with a user study and a limited analytical study where they replicate only one IR-based technique [8]. In order to gather more strong empirical evidences, we replicate three IR-based bug localization techniques– BugLocator [8], BLUiR [6] and LOBSTER [3], and conduct experiments using 5,500 bug reports [1] from eight open source systems. We answer the following research question through our study:

RQ: How do existing IR-based techniques perform with the bug reports containing (a) excessive structured information (e.g., stack traces), and (b) no structured information (i.e., only regular texts)?

2.1 Study Design

Dataset Collection: We collect RESOLVED bug reports from either BugZilla or JIRA repository of each subject system, extract *changeset* (i.e., list of changed files) from their corresponding bug-fixing commits at GitHub, and then develop bug report-solution pairs [2]. We also divide our report collection into three clusters– BR$_{NL}$, BR$_{ST}$ and BR$_{PE}$–based on the extent or type of structured information they have [7]. Each bug report from BR$_{NL}$ contains only natural language texts whereas a report from BR$_{ST}$ contains one or more stack traces besides other structured entities. On the contrary, each bug report from BR$_{PE}$ contains regular texts, and one or more program entity names (e.g., method names) but no stack traces. We use a semi-automated approach in the report clustering where regular expressions and manual analysis were applied.

Variables of Study: We collect authors' implementation of BugLocator and BLUiR from their online sources, and replicate LOBSTER ourselves which is verified using the authors' provided experimental data. We choose three *independent variables* involving bug report quality, retrieval engine and text preprocessing step and one *response variable* involving bug localization performance. We adopt a systematic approach using these variables and choose our baseline approach for bug localization. In short, the baseline uses *whole texts* (i.e., preprocessed without stemming) of a bug report as a *query*, and *Lucene* as the *text retrieval engine.*

2.2 Results and Discussions

Impact of Excessive Structured Information: We conduct experiments using the bug reports that contain stack traces (i.e., BR$_{ST}$), evaluate four techniques (i.e., three existing, one baseline), and report our findings. We found that *BugLocator* performed the best among all four techniques in terms of Hit@K, Mean Reciprocal Rank@K, and Mean Average Precision@K. From Fig. 1, we see that BugLocator achieves ≈40% mean average precision for various Top-K results. However, precision measures of each technique (especially BLUiR and LOBSTER) for BR$_{ST}$ are significantly lower

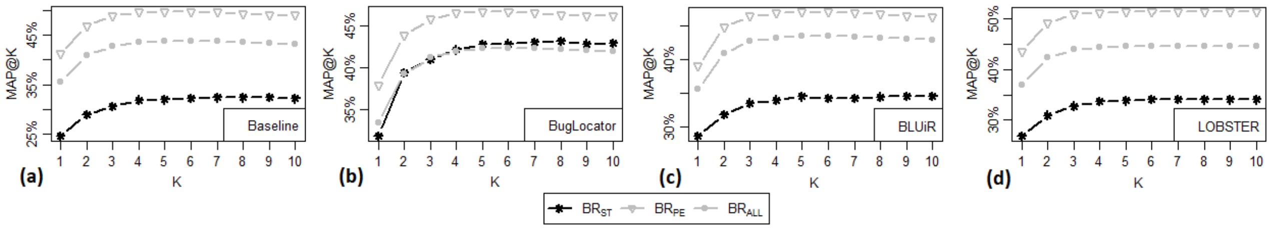

Figure 1: MAP@K of (a) Baseline, (b) BugLocator, (c) BLUiR, and (d) LOBSTER with bug reports containing stack traces (i.e., BR$_{ST}$)

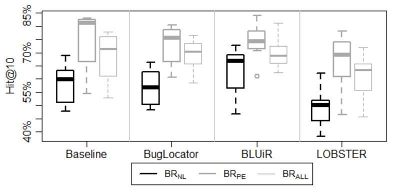

Figure 2: Hit@10 of Baseline, BugLocator, BLUiR, and LOBSTER with bug reports containing only NL texts (i.e., BR$_{NL}$)

(i.e., all *p-values*<0.05 and Cliff's $0.82 \leq \Delta \leq 1.00$) than that for BR$_{PE}$ (i.e., reports containing program entities) or BR$_{ALL}$ (i.e., all bug reports). That is, the same technique performs significantly lower than the average (i.e., for BR$_{ALL}$) when the reports contain stack traces. Such finding supports the conjecture about negative impacts of noisy queries on the IR-based bug localization [7], and thus also warrants for query refinement prior to the bug localization.

Impact of the Absence of Structured Information: We also conduct experiments with the bug reports that contain only natural language texts but no structured entities (i.e., BR$_{NL}$), and evaluate four techniques under study. We found that *BLUiR* performed the best in terms of several performance metrics. Fig. 2 shows the Hit@10 of all techniques for BR$_{NL}$ dataset. It is interesting to note that the Hit@10 of BLUiR is significantly higher (i.e., *p-value*=0.02<0.05, Δ=0.50 *(large)*) than that of BugLocator although BugLocator performed the best with BR$_{ST}$ dataset. Furthermore, no existing technique except BLUiR strangely performs better than the baseline for this dataset, BR$_{NL}$, which is a bit counter-intuitive. Such finding might explain the role of underlying text retrieval engines given that each of the four techniques used the same query (i.e., from BR$_{NL}$) for the bug localization. However, it also should be noted that each technique performed significantly lower (i.e., *p-values*<0.05 and $0.63 \leq \Delta \leq 0.84$ (*large*)) for BR$_{NL}$ than their average performance (i.e., for BR$_{ALL}$) irrespective of their back-end retrieval engines. That is, the queries were possibly not good enough for the localization due to their lack of relevant structured information, and thus, they need meaningful expansions.

From Figures 1, 2, we also see that no existing technique is robust enough against either collection – BR$_{ST}$ or BR$_{NL}$. While BugLocator performs well for BR$_{ST}$, it does poorly for BR$_{NL}$. The opposite is true for BLUiR. On the other hand, every technique performs exceptionally well (i.e., 70%-80% median Hit@10) with the high quality bug reports, i.e., BR$_{PE}$. More interestingly, even the baseline Hit@10 is higher than that of all three competing techniques. All these findings above suggest that quality dynamics of a bug report is a major aspect of IR-based bug localization which was possibly overlooked by the earlier approaches. One could also argue that most of the existing IR-based techniques [4–6, 8] employ the whole report texts (i.e., title and description) without major modifications as a *query* for bug localization. Hence, their query could be either noisy due to excessive structured information (e.g., stack traces) or insufficient due to the lack of relevant structured information (e.g., program entity names). Thus, appropriate reformulations should be applied to the queries especially generated from BR$_{ST}$ and BR$_{NL}$ report categories before using them for bug localization with information retrieval.

3 RELATED WORK

Information Retrieval (IR)-based bug localization has been an active area of research for a while. Several studies employ traditional IR methods such as Latent Semantic Analysis (LSA), Latent Dirichlet allocation (LDA) [4] and Vector Space Model (VSM) [3, 6, 8] in the bug localization, and they are found to be cheap and effective [5]. Unfortunately, recent findings [7] suggest two inherent limitations of the IR-based localization approaches. They are greatly affected by the (low) quality of the bug reports, i.e., queries. Our work in this paper has empirically re-investigated such qualitative observations using a much larger dataset and has confirmed their validity.

4 CONCLUSION & FUTURE WORK

Existing IR-based bug localization techniques are significantly affected both by the overwhelming presence (e.g., stack traces) and by the total absence of structured entities (i.e., only regular texts) in the bug reports, i.e., queries. In fact, no existing technique is robust enough against either of these two types of reports simultaneously. Future works should incorporate quality dynamics of bug reports and query reformulations into the IR-based bug localization in order to overcome the challenges outlined by this study.

ACKNOWLEDGEMENT

This research is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), and by a Canada First Research Excellence Fund (CFREF) grant coordinated by the Global Institute for Food Security (GIFS) at University of Saskatchewan.

REFERENCES

[1] 2018. Empirical Dataset. (2018). http://www.usask.ca/~mor543/icse2018
[2] A. Bachmann and A. Bernstein. 2009. Software Process Data Quality and Characteristics: A Historical View on Open and Closed Source Projects. In *Proc. IWPSE*. 119–128.
[3] L. Moreno, J. J. Treadway, A. Marcus, and W. Shen. 2014. On the Use of Stack Traces to Improve Text Retrieval-Based Bug Localization. In *Proc. ICSME*. 151–160.
[4] A. T. Nguyen, T. T. Nguyen, J. Al-Kofahi, H. V. Nguyen, and T. N. Nguyen. 2011. A Topic-based Approach for Narrowing the Search Space of Buggy Files from a Bug Report. In *Proc. ASE*. 263–272.
[5] S. Rao and A. Kak. 2011. Retrieval from Software Libraries for Bug Localization: A Comparative Study of Generic and Composite Text Models. In *Proc. MSR*. 43–52.
[6] R. K. Saha, M. Lease, S. Khurshid, and D. E. Perry. 2013. Improving bug localization using structured information retrieval. In *Proc. ASE*. 345–355.
[7] Q. Wang, C. Parnin, and A. Orso. 2015. Evaluating the Usefulness of IR-based Fault Localization Techniques. In *Proc. ISSTA*. 1–11.
[8] J. Zhou, H. Zhang, and D. Lo. 2012. Where should the bugs be fixed? More accurate information retrieval-based bug localization based on bug reports. In *Proc. ICSE*. 14–24.