Retrieval from software libraries for bug localization: A comparative study of generic and composite text models - A Retrospective

Shivani Rao, LinkedIn
raoshivani@gmail.com

Avinash Kak, Purdue University
kak@purdue.edu

ABSTRACT

This retrospective on our 2011 MSR publication starts with the research milieu that led to the work reported in our paper. We briefly review the competing ideas of a decade ago that could be applied to solving the problem of identifying the files in a software library related to a query. We were especially interested in finding out if the more complex text retrieval methods of that time would be effective in the software context. A surprising conclusion of our paper was that the reality was exactly the opposite: the more traditional simpler methods outperformed the complex methods. In addition to this surprising result, our paper was also the first to report what was considered at that time a large-scale quantitative evaluation of the IR-based approaches to automatic bug localization. Over the years, such quantitative evaluations have become the norm. We believe that these contributions were largely responsible for the popularity of this paper in the research literature.

1. INCEPTION

In 2008, Shivani Rao started her PhD at Purdue University and in Fall of 2008 joined Avinash Kak's Robot Vision Lab for her graduate research. Her project was part of a larger research program at Purdue that was funded by Infosys through its SETLabs (Software Engineering & Technology). The Infosys SETlabs focused on improving developer effectiveness with innovative tools. These tools revolved around making sense of and managing the complexity of large-scale software systems using tools such as model-driven software development, software modularization, program comprehension, and so on.

2. SETTING THE SCENE

During that time in the research community, there was increasing interest in applying Data Mining and Machine Learning techniques to solving problems in software engineering. In order to bring these ideas together, a workshop called "Mining of Software Repositories (MSR)" was organized in 2004 as a co-located venue under the umbrella of ICSE that year. By 2008, the research interest in the field had grown to an extent that the MSR workshop had now turned into a regular conference in its own right. For the most part, the approaches presented at the MSR venues had a common theme: Treat source code as documents containing text and apply machine learning and text understanding approaches to solving software engineering problems that included duplicate bug detection, bug localization, program comprehension, and others.

Our own research at that time focused on the problem of *bug localization*, meaning identifying the files that would need to be looked at in response to a bug report. The previous approaches to bug localization could be broadly categorized as falling into static and dynamic methods. Static bug localization techniques operated by examining a file against a set of rules that predicted if the code was buggy. On the other hand, dynamic bug localization techniques relied on comparing the control flows for the passing and the failing runs to identify the location of a bug. As a large departure from that prior art, we started to investigate the applicability of IR (Information Retrieval) based approaches for solving the problem. We believed that such approaches would allow the programmers to locate and fix the bugs faster. In IR based approaches, the bugs are treated as queries and the source code as a corpus of documents to be searched in response to a query. With IR, the returned result is a ranked list of the files relevant to a query in decreasing order of relevancy.

3. OUR OVERARCHING GOAL

Around the time we started exploring the application of IR to bug localization, the literature generally entailed small query sets (consisting of, say, 5-15 queries) and the conclusions in these studies were mostly qualitative in nature. Additionally, different researchers did their analyses on different repositories, or with different models, making it difficult to carry out a side-by-side comparison of the solutions presented. That led to the following overarching goals for our research:

- To compare the state-of-art text models used in IR on the same set of repositories;
- To carry out a large scale quantitative study with a standardized dataset so that other researchers could utilize the same dataset for taking forward our work.

4. CONDUCT

Within the text understanding community, there was a lot of buzz at that time about the LDA (Latent Dirichlet Allocation) approach for representing documents. Compared to the other models of that time period, LDA and its offshoots were the most complex and they entailed a hidden layer of "topic" variables, with each topic being represented as a probability distribution over the words in the vocabulary of the corpus. We decided to compare this model with four much simpler models for representing the documents: Unigram, VSM (Vector Space model), LSA (Latent Semantic Analysis) and CBDM (Cluster based Document Model). We also created variants of these models to better understand their relative strengths and weaknesses.

4.1 Collection and Analyzing the data

We used a benchmark dataset, iBugs, which, at that time, was popular for studying static and dynamic bug localization techniques.

Since iBugs had never been used previously for IR based bug localization, our work included the data conditioning steps needed to

represent the source files and the bug reports as documents. These data conditioning steps included eliminating common words from the vocabulary that would not be discriminative (e.g. words such as 'for', 'while', 'with', 'each', etc), splitting camel-cased and hyphenated words that are frequently used as variable names in source code, eliminating unicode strings, and so on.

4.2 Metrics

We used the following metrics to compare the different algorithms: MAP (Mean Average Precision) and SCORE (Rank of Retrieved Files). By definition, MAP ranges between 0 and 1 and can be interpreted as an average measure of the proportion of the returned files relevant to a query. And, the SCORE@R tells us how many bugs (or queries) would be correctly located if we only examined the files returned up to rank R. So, SCORE@1 tells us how many bugs would be correctly located if we just looked at the first source file in the list returned by the algorithm.

4.3 Findings

In general, one expects the more complex models to handle more difficult data conditions. At the time of our research, LDA was new and a lot of academic researchers had jumped into the LDA bandwagon. Going into our comparative evaluation, we had fully expected the LDA modeling approach to significantly outperform the other competing approaches. Just imagine our own surprise when the results turned out to be exactly the opposite. The conclusion of our evaluation was that the simplest of the models outperformed the more complex ones like LDA. What makes this story even more interesting is that even the just moderately complex text models like LSI and CBDM did not outperform the simpler ones like VSM or Unigram.

5. FURTHER WORK

Our work laid the foundations for several other contributions that subsequently emerged from our lab. One such early contribution was our collaboration with Emily Hill that was devoted to studying the effectiveness of the different stemming algorithms in the software context [Hill et al. 2012]. We also investigated incremental update frameworks for IR based bug localization, the goal here being to make incremental updates to the model so that it could evolve with changes to the repository. We analyzed the different popular models for their incremental update versions. This work was done in collaboration with Henry Medeiros ([Rao et al. 2013], [Rao et al. 2015]).

6. IMPACT

To assess the impact of our work on the broader research community, we reviewed the literature that has cited our paper, focusing especially on those publications that have been cited more than 50 times. The articles we reviewed were published in a wide range of high-impact venues such as MSR, IEEE TSE, ICSE, ASE, and others. While some of these articles surveyed the latest state-of-the-art in IR based approaches ([Wong et al. 2016], [Chen et al. 2015], [Zhang et al. 2015], [Hemmati et al. 2013]), with regard to the others, broadly speaking, those fell in two distinct categories: those that proposed new improvements to the IR based bug localization techniques, and those that either further affirmed our own conclusions or applied the IR-based tools to other problems/domains. Here are some examples of the papers in the first category:

- In ([Zhou et al. 2012], [Saha et al. 2013], [Wang and Lo 2014], [Wong et al. 2014], [Wang and Lo 2014], [Sisman and Kak 2012], [Sisman and Kak 2013], [Sisman et al. 2017]) the authors have proposed approaches that are unsupervised and improve the retrieval accuracy by augmenting the document representation or the query representation with additional sources of information.

- In ([Ye et al. 2014], [Kim et al. 2013]), the authors have proposed supervised approaches for retrieval (Learning to Rank, SVMs etc) that partitioned the data into training and testing sets and, subsequently, trained either a classification or a regression model to improve the retrieval accuracy.

- In contrast with the papers cited above that focus on developing supervised and unsupervised methods, the authors of ([Le et al. 2015], [Le et al. 2016]) have proposed novel integrated methods that combine IR based algorithms with those based on dynamic bug localization.

- More recently, the authors of ([Ye et al. 2016], [Lam et al. 2017], [Akbar and Kak 2019]) have demonstrated how word embeddings produced by deep-learning based algorithms like word2vec can be used to improve retrieval accuracy.

And here are the more prominent papers in the second category:

- [Wang et al. 2011] replicated our work in a different application and came to the same overall conclusions as ours. They concluded that it was not only the vanilla LDA models that failed to outperform the much simpler VSM, even the more sophisticated version of LDA, like the Hierarchical LDA, and the non-negative matrix factorization method could not beat VSM.

- About using IR based search tools for solving other problems in Software Engineering, [Saha et al. 2015] proposed using such tools for regression testing; [Thung et al. 2013] showed how such tools can be used to recommend API methods in response to feature requests; [Wen et al. 2016], [Yang et al. 2014], and [Zhang et al. 2016] suggested using tools for bug triaging that involves finding the right developer for fixing a bug and assigning a severity level to the bug.

- The rest of the papers we reviewed in this category include those that have investigated improvements to the topic models for improving retrieval accuracy ([Chen et al. 2015], [Agrawal et al. 2018] , [Biggers et al. 2014]) and a paper that has questioned the use of IR-based tools for bug localization [Wang et al. 2015].

7. CONCLUSIONS

This retrospective provides a context for the work that was reported in our paper, highlighting the popularity of certain IR algorithms of that era. The buzz associated with those algorithms was our primary motivation for investigating their effectiveness in the software context. As it turned out, our work demonstrated that the actual performance of those algorithms was inversely proportional to their complexity; a finding that was later corroborated by other researchers. Additionally, our paper set the norm of performing large-scale quantitative evaluation of the IR-based approaches to automatic bug localization. These were the reasons for the popularity of our paper and its high citation count. In this retrospective, we have also surveyed some of the more notable contributions that followed ours and that have cited out work.

8. REFERENCES

[Agrawal et al. 2018] Amritanshu Agrawal, Wei Fu, and Tim Menzies. 2018. What is Wrong with Topic Modeling? And How to Fix it Using Search-based Software Engineering. *Information and Software Technology* 98 (2018), 74–88.

[Akbar and Kak 2019] S. Akbar and A. Kak. 2019. SCOR: Source Code Retrieval with Semantics and Order. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. 1–12.

[Biggers et al. 2014] Lauren R Biggers, Cecylia Bocovich, Riley Capshaw, Brian P Eddy, Letha H Etzkorn, and Nicholas A Kraft. 2014. Configuring Latent Dirichlet Allocation Based Feature Location. *Empirical Software Engineering* 19, 3 (2014), 465–500.

[Chen et al. 2015] Tse-Hsun Chen, Stephen Thomas, and Ahmed E. Hassan. 2015. A Survey on the use of Topic Models when Mining Software Repositories. 21 (2015).

[Hemmati et al. 2013] H. Hemmati, S. Nadi, O. Baysal, O. Kononenko, W. Wang, R. Holmes, and M. W. Godfrey. 2013. The MSR Cookbook: Mining a Aecade of Research. In *2013 10th Working Conference on Mining Software Repositories (MSR)*. 343–352.

[Hill et al. 2012] E. Hill, S. Rao, and A. Kak. 2012. On the Use of Stemming for Concern Location and Bug Localization in Java. In *2012 IEEE 12th International Working Conference on Source Code Analysis and Manipulation*. 184–193.

[Kim et al. 2013] D. Kim, Y. Tao, S. Kim, and A. Zeller. 2013. Where Should We Fix This Bug? A Two-Phase Recommendation Model. *IEEE Transactions on Software Engineering* 39, 11 (2013), 1597–1610.

[Lam et al. 2017] A. N. Lam, A. T. Nguyen, H. A. Nguyen, and T. N. Nguyen. 2017. Bug Localization with Combination of Deep Learning and Information Retrieval. In *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)*. 218–229.

[Le et al. 2016] Tien-Duy B. Le, David Lo, Claire Le Goues, and Lars Grunske. 2016. A Learning-to-Rank Based Fault Localization Approach Using Likely Invariants. In *Proceedings of the 25th International Symposium on Software Testing and Analysis* (Saarbrücken, Germany) *(ISSTA 2016)*. Association for Computing Machinery, New York, NY, USA, 177–188.

[Le et al. 2015] Tien-Duy B. Le, Richard J. Oentaryo, and David Lo. 2015. Information Retrieval and Spectrum Based Bug Localization: Better Together. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering* (Bergamo, Italy) *(ESEC/FSE 2015)*. Association for Computing Machinery, New York, NY, USA, 579–590.

[Rao et al. 2013] S. Rao, H. Medeiros, and A. Kak. 2013. An Incremental Update Framework for Efficient Retrieval from Software Libraries for Bug Localization. In *2013 20th Working Conference on Reverse Engineering (WCRE)*. 62–71.

[Rao et al. 2015] Shivani Rao, Henry Medeiros, and Avinash Kak. 2015. Comparing Incremental Latent Semantic Analysis Algorithms for Efficient Retrieval from Software Libraries for Bug Localization. *SIGSOFT Softw. Eng. Notes* 40, 1 (Feb. 2015), 1–8.

[Saha et al. 2013] R. K. Saha, M. Lease, S. Khurshid, and D. E. Perry. 2013. Improving Bug Localization Using Structured Information Retrieval. In *2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 345–355.

[Saha et al. 2015] R. K. Saha, L. Zhang, S. Khurshid, and D. E. Perry. 2015. An Information Retrieval Approach for Regression Test Prioritization Based on Program Changes. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 1. 268–279.

[Sisman et al. 2017] Bunyamin Sisman, Shayan A. Akbar, and Avinash C. Kak. 2017. Exploiting Spatial Code Proximity and Order for Improved Source Code Retrieval for Bug Localization. *Journal of Software: Evolution and Process* 29, 1 (2017), e1805. e1805 JSME-16-0104.R1.

[Sisman and Kak 2012] B. Sisman and A. C. Kak. 2012. Incorporating Version Histories in Information Retrieval Based Bug Localization. In *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*. 50–59.

[Sisman and Kak 2013] B. Sisman and A. C. Kak. 2013. Assisting Code Search with Automatic Query Reformulation for Bug Localization. In *2013 10th Working Conference on Mining Software Repositories (MSR)*. 309–318.

[Thung et al. 2013] F. Thung, S. Wang, D. Lo, and J. Lawall. 2013. Automatic Recommendation of API Methods from Feature Requests. In *2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 290–300.

[Wang et al. 2015] Qianqian Wang, Chris Parnin, and Alessandro Orso. 2015. Evaluating the Usefulness of IR-Based Fault Localization Techniques *(ISSTA 2015)*. Association for Computing Machinery, New York, NY, USA, 1–11.

[Wang and Lo 2014] Shaowei Wang and David Lo. 2014. Version History, Similar Report, and Structure: Putting Them Together for Improved Bug Localization. In *Proceedings of the 22nd International Conference on Program Comprehension* (Hyderabad, India) *(ICPC 2014)*. Association for Computing Machinery, New York, NY, USA, 53–63.

[Wang et al. 2011] S. Wang, D. Lo, Z. Xing, and L. Jiang. 2011. Concern Localization using Information Retrieval: An Empirical Study on Linux Kernel. In *2011 18th Working Conference on Reverse Engineering*. 92–96.

[Wen et al. 2016] M. Wen, R. Wu, and S. Cheung. 2016. Locus: Locating Bugs from Software Changes. In *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 262–273.

[Wong et al. 2014] C. Wong, Y. Xiong, H. Zhang, D. Hao, L. Zhang, and H. Mei. 2014. Boosting Bug-Report-Oriented Fault Localization with Segmentation and Stack-Trace Analysis. In *2014 IEEE International Conference on Software Maintenance and Evolution*. 181–190.

[Wong et al. 2016] W. E. Wong, R. Gao, Y. Li, R. Abreu, and F. Wotawa. 2016. A Survey on Software Fault Localization. *IEEE Transactions on Software Engineering* 42, 8 (2016), 707–740.

[Yang et al. 2014] G. Yang, T. Zhang, and B. Lee. 2014. Towards Semi-automatic Bug Triage and Severity Prediction Based on Topic Model and Multi-feature of Bug Reports. In *2014 IEEE 38th Annual Computer Software and Applications Conference*. 97–106.

[Ye et al. 2014] Xin Ye, Razvan Bunescu, and Chang Liu. 2014. 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* (Hong Kong, China) *(FSE 2014)*. Association for Computing Machinery, New York, NY, USA, 689–699.

[Ye et al. 2016] Xin Ye, Hui Shen, Xiao Ma, Razvan Bunescu, and Chang Liu. 2016. From Word Embeddings to Document Similarities for Improved Information Retrieval in Software Engineering *(ICSE '16)*. Association for Computing Machinery, New York, NY, USA, 404–415.

[Zhang et al. 2015] Jie Zhang, Xiaoyin Wang, Dan Hao, Bing Xie, Lu Zhang, and Hong Mei. 2015. A Survey on Bug-Report Analysis. *Science China Information Sciences* 58, 2 (2015), 1–24.

[Zhang et al. 2016] Tao Zhang, Jiachi Chen, Geunseok Yang, Byungjeong Lee, and Xiapu Luo. 2016. Towards More Accurate Severity Prediction and Fixer Recommendation of Software Bugs. *Journal of Systems and Software* 117 (2016), 166–184.

[Zhou et al. 2012] 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 *2012 34th International Conference on Software Engineering (ICSE)*. 14–24.