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


