

ASE 2024 Most Influential Paper (MIP) Talk



Improving Bug Localization using Structured Retrieval

Ripon Saha, Matthew Lease, Sarfraz Khurshid, Dewayne Perry [ASE 2013]

Ripon Saha
Meta

Sarfraz Khurshid
The University of Texas at Austin



Dr. Ripon Saha



Prof. Matt Lease



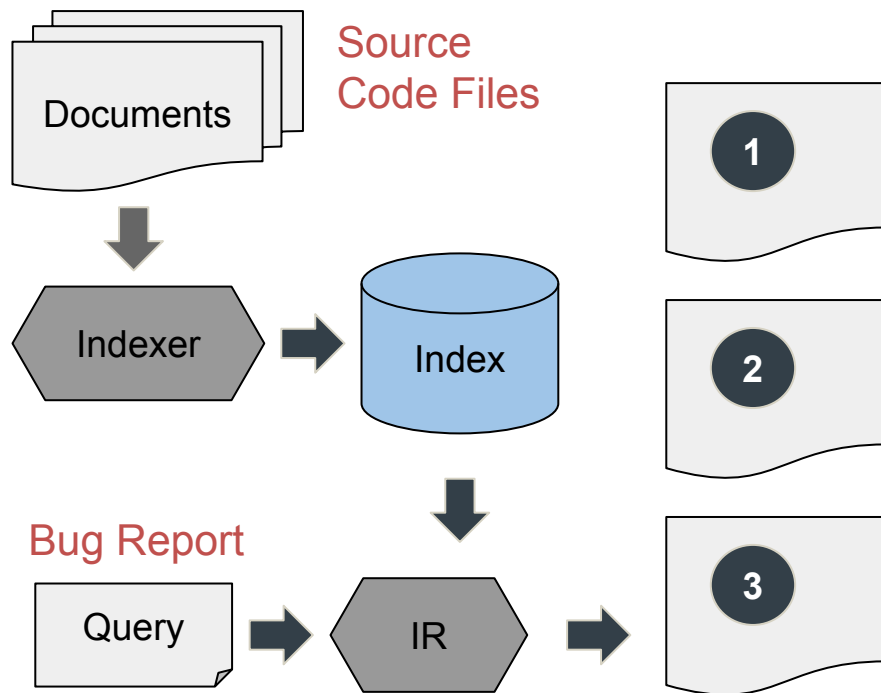
Prof. Sarfraz Khurshid



Prof. Dewayne Perry

Background

- ❑ In 2013, global cost of software debugging was \$312/year
- ❑ Average time to fix a bug is 6 months in both commercial and open-source projects
- ❑ Spectrum-based fault localization was common
- ❑ The exploration of Information Retrieval (IR)-based bug localization had started and shown to hold promise



A general IR system

Prior work: Structured Retrieval (1/2)

First ACM international conference on Digital libraries, 1996



Index Structures for Structured Documents *

Yong Kyu Lee, Seong-Joon Yoo, Kyoungro Yoon
School of Computer and Information Science
Syracuse University

P. Bruce Berra
Dept. of Electrical and Computer Engineering
Syracuse University

Abstract

Much research has been carried out in order to manage structured documents such as SGML documents and to provide powerful query facilities which exploit document structures as well as document contents. In order to perform structure queries efficiently in a structured document management system, an index struc-

logical structure of documents such as chapters, sections, or subsections. The structure query can be combined with the content query to build a more powerful query.

Much research has been performed to design efficient index structures for database and information retrieval systems [7] [10] [11] [12] [14] [24] [25]. In order to perform structure queries efficiently in the structured

Prior work: Structured Retrieval (2/2)

Computer networks and ISDN systems, 1998 - Elsevier

The Anatomy of a Large-Scale Hypertextual Web Search Engine

Sergey Brin and Lawrence Page

*Computer Science Department,
Stanford University, Stanford, CA 94305, USA*
sergey@cs.stanford.edu and page@cs.stanford.edu

Abstract

In this paper, we present Google, a prototype of a large-scale search engine which makes heavy use of the structure present in hypertext. Google is designed to crawl and index the Web efficiently and produce much more satisfactory search results than competing systems. The prototype with a full text and hyperlink database has been running for over a year. The search engine is available at <http://google.stanford.edu/>. To engineer a search engine that handles tens to hundreds of millions of web pages involving a comparable number of distinct terms. They answer tens of millions of queries every day. Despite the importance of large-scale search engines on the web, very little academic research has been done on them. Furthermore, due to rapid advance in

*heavy use of the structure
present in hypertext*

Idea: Code has structure!

Bug ID: 80720

Summary: *Pinned console does not remain on top*

Description: *Open two console views, ... Pin one console. Launch another program that produces output. Both consoles display the last launch. The pinned console should remain pinned.*

Source code file: **ConsoleView.java**

```
public class ConsoleView extends PageBookView
implements IConsoleView, IConsoleListener {...
    public void display(IConsole console) {
        if (fPinned && fActiveConsole != null){return;}
    }...
    public void pin(IConsole console) {
        if (console == null) {
            setPinned(false);
        } else {
            if (isPinned()) {
                setPinned(false);
            }
            display(console);
            setPinned(true); ..
        }
    }
}
```

Both bug report and document have rich structure!

Idea: Constructs Hold Valuable Info

Bug ID: 80720

Summary: *Pinned console does not remain on top*

Description: *Open two console views, ... Pin one console. Launch another program that produces output. Both consoles display the last launch. The pinned console should remain pinned.*

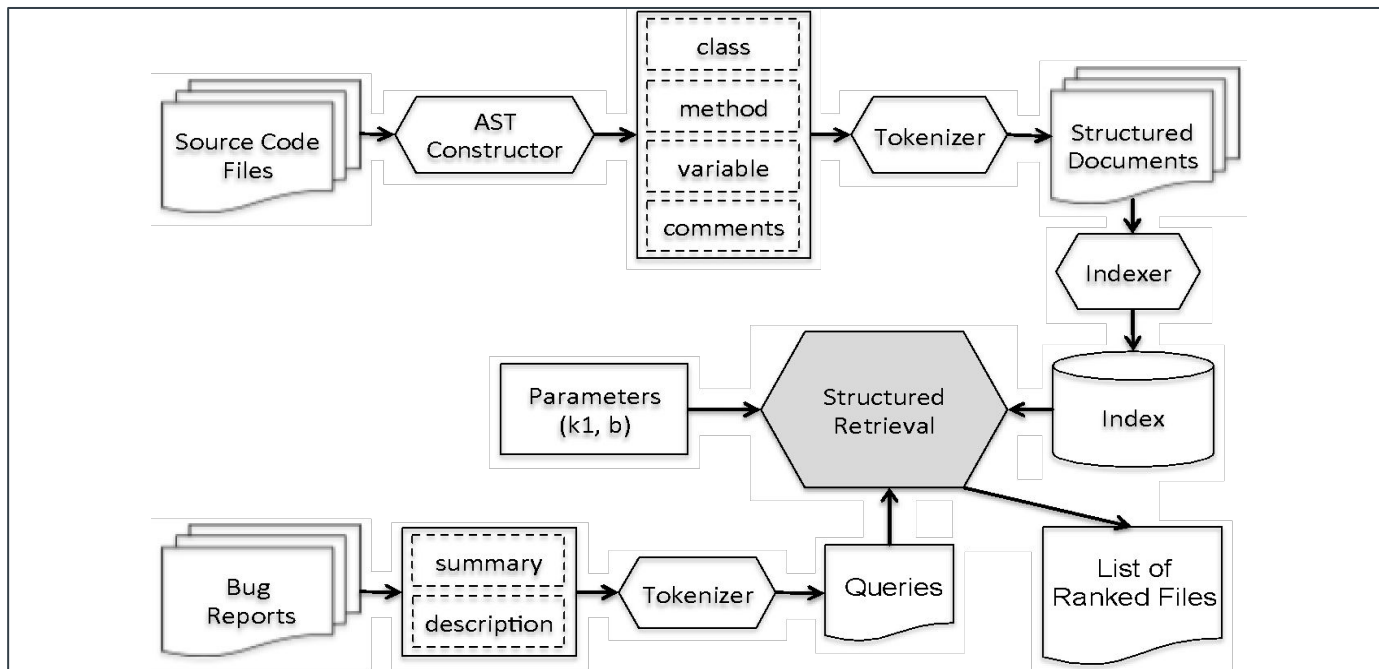
Figure: A real bug report from Eclipse Project and the corresponding fixed source code [Zhou et al., ICSE 2012]

Source code file: **ConsoleView.java**

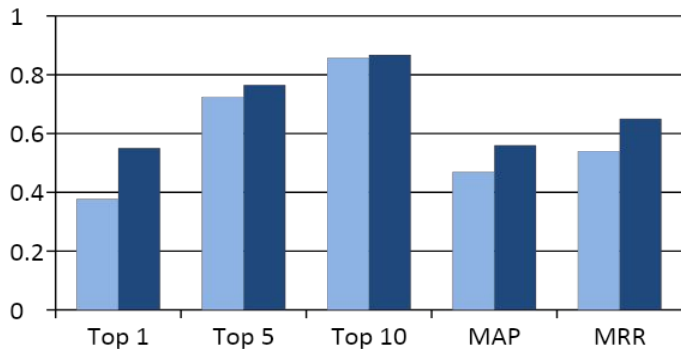
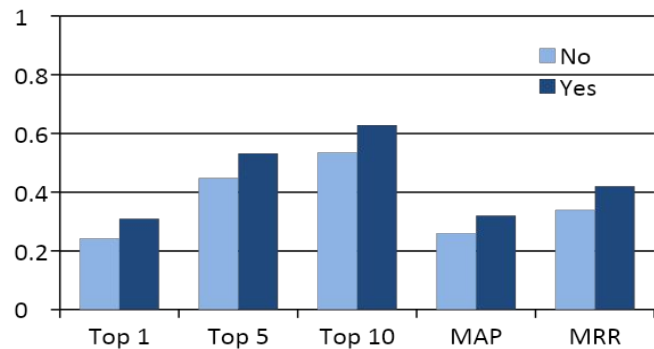
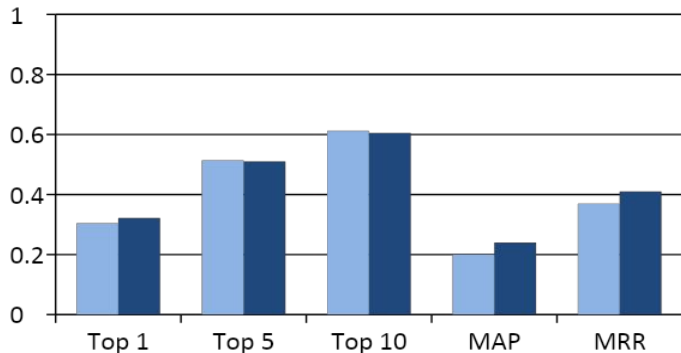
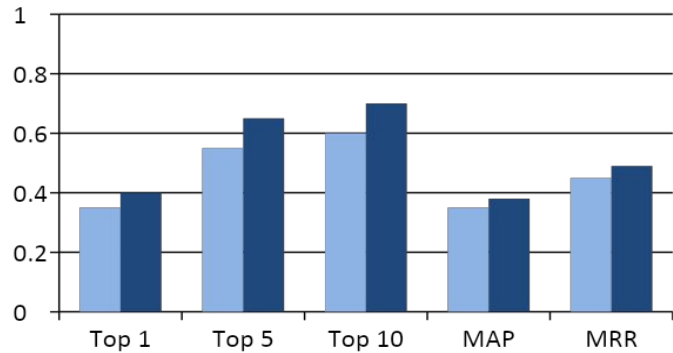
```
public class ConsoleView extends PageBookView
implements IConsoleView, IConsoleListener {...
    public void display(IConsole console) {
        if (fPinned && fActiveConsole != null){return;}
    }...
    public void pin(IConsole console) {
        if (console == null) {
            setPinned(false);
        } else {
            if (isPinned()) {
                setPinned(false);
            }
            display(console);
            setPinned(true);
        }
    }
}
```

Higher level of program constructs have more important information

Our System Architecture



Improvement from Leveraging Structure

SWT

Eclipse

AspectJ

ZXing


Titles from all Citing Papers



Impact: SE Areas Influenced

Areas	# Citations
Fault Localization	234
Other Bug Related	99
IR in SE	15
Recommendation	14
Tests	12
Security	11
Feature Location, API, Program Repair	20

Conferences/Journal	# Citations
TSE	27
ICSE	26
arXiv	25
ASE	19
IST	14
JSS	13
FSE	10

Non-SE: AAI, USENIX, S&P, ICDM etc.

Impact from Proposed Future Work

In our future research, we would like to explore the following areas to further improve our model: bug report summarization and learning parameters.

Bug Report Summarization. In this paper, we showed how the performance of bug localization improves by focusing on condensed information such as bug summaries, class names, or method names. However, we still used exactly the same long bug descriptions from bug reports. There are some automatic techniques [22] that can condense bug descriptions up to 30% of its original size. Such summarized bug descriptions may further improve the performance of bug localization.

Improving IR-Based Bug Localization with Context-Aware Query Reformulation

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

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

ABSTRACT

Recent findings suggest that Information Retrieval (IR)-based bug localization techniques do not perform well if the bug report lacks rich structured information (e.g., relevant program entity names). Conversely, excessive structured information (e.g., stack traces) in the bug report might not always help the automated localization either. In this paper, we propose a novel technique—BUZZARD—that

three steps of debugging is the identification of the location of a bug in the source code, i.e., bug localization [37, 56]. Recent bug localization techniques can be classified into two broad families—*spectra based* and *information retrieval (IR) based* [29]. While spectra-based techniques rely on execution traces of a software system, IR-based techniques analyse shared vocabulary between a bug report (i.e., query) and the project source for bug localization [34, 66]. Perform-


2018

Soft Computing (2021) 25:7307–7323
<https://doi.org/10.1007/s00500-021-05689-2>

METHODOLOGIES AND APPLICATION



On the classification of bug reports to improve bug localization

Fan Fang¹ · John Wu¹ · Yanyan Li¹ · Xin Ye¹ · Wajdi Aljedaani² · Mohamed Wiem Mkaouer³ 

Accepted: 11 February 2021 / Published online: 19 March 2021
 © The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2021

Abstract

Bug localization is the automated process of finding the possible faulty files in a software project. Bug localization allows

Impact from Proposed Future Work

In our future research, we would like to explore the following areas to further improve our model: bug report summarization and learning parameters.

Bug Report Summarization. In this paper, we showed how the performance of bug localization improves by focusing on condensed information such as bug summaries, class names, or method names. However, we still used exactly the same long bug descriptions from bug reports. There are some automatic techniques [22] that can condense bug descriptions up to 30% of its original size. Such summarized bug descriptions may further improve the performance of bug localization.

Learning to Rank. To tune the value of k_1 and b in our model, we ran BLUiR on AspectJ using a range of values at a fixed interval length and took the pair for which we got the best result for other subject systems. However, the best values may be different for different subject systems. Finding a globally optimal weights is still an open problem in IR research community. Recent work [20] in IR is using machine learning methods to automatically optimize ranking parameters for more sophisticated ranking functions. This would provide another interesting direction for future studies.

Learning to Rank Relevant Files for Bug Reports using Domain Knowledge

Xin Ye, Razvan Bunescu, and Chang Liu
School of Electrical Engineering and Computer Science, Ohio University
Athens, Ohio 45701, USA
xy348709,bunescu,liuc@ohio.edu

ABSTRACT

When a new bug report is received, developers usually need to reproduce the bug and perform code reviews to find the cause, a process that can be tedious and time consuming. A tool for ranking all the source files of a project with respect to how likely they are to contain the cause of the bug would enable developers to narrow down their search and potentially could lead to a substantial increase in productivity. This paper introduces an adaptive ranking approach that leverages domain knowledge through functional decompositions of source code files into methods, API descriptions of library components used in the code, the bug-fixing history, and the code change history. Given a bug report, the ranking score of each source file is computed as a weighted combination of an array of features encoding domain knowledge, where the weights are trained automatically on previously solved bug reports using a learning-to-rank technique. We evaluated our system on six large scale open source Java projects, using the before-fix version of the project for every bug report. The experimental results show that the newly introduced learning-to-rank approach significantly outperforms two recent state-of-the-art methods in recommending relevant files for bug reports. In particular, our method makes correct recommendations within the top 10 ranked source files for over 70% of the bug reports in the Eclipse Platform and Tomcat projects.

Categories and Subject Descriptors

Keywords

bug reports, software maintenance, learning to rank

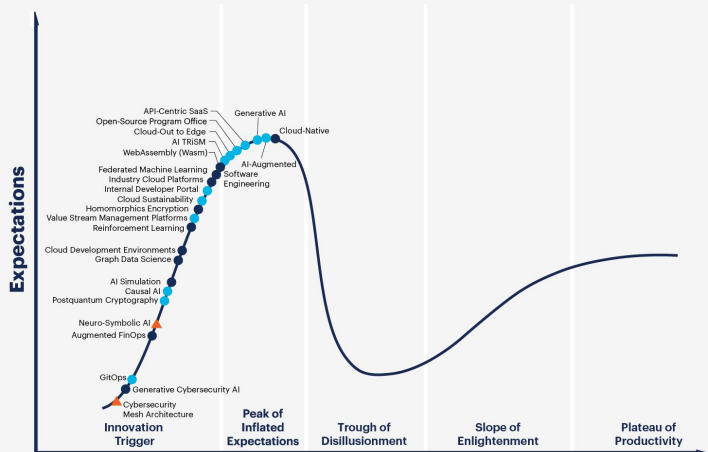
1. INTRODUCTION AND MOTIVATION

A software *bug* or *defect* is a coding mistake that may cause unintended and unexpected behaviors of the software component [7]. Upon discovering an abnormal behavior of the software project, a developer or a user will report it in a document, called a *bug report* or *issue report*. A bug report provides information that could help in fixing a bug, with the overall aim of improving the software quality. A large number of bug reports could be opened during the development life-cycle of a software product. For example, there were 3,389 bug reports created for the Eclipse Platform product in 2013 alone. In a software team, bug reports are extensively used by both managers and developers in their daily development process [10].

A developer who is assigned a bug report usually needs to reproduce the abnormal behavior [22] and perform code reviews [2] in order to find the cause. However, the diversity and uneven quality of bug reports can make this process nontrivial. Essential information is often missing from a bug report [6]. Lexical mismatches between natural language statements in bug reports and technical terms in software systems [4] limit the accuracy of ranking methods that are based on simple lexical matching scores. To locate the bug, a developer needs to not only analyze the bug report using their domain knowledge, but also collect information from peer developers and users. Employing such a manual process

Christopher Manning, SIGIR 2016 Talk

Hype Cycle for Emerging Technologies, 2023



Plateau will be reached:
 ○ less than 2 years ● 2 to 5 years ● 5 to 10 years ▲ more than 10 years As of August 2023

gartner.com

Source: Gartner
 © 2023 Gartner, Inc. and/or its affiliates. All rights reserved. 2079700

Neural models have transformed IR & NLP fields since our work



2011
speech

2013
vision

2015
NLP

2017
IR

Onal et al. Neural Information Retrieval: At the End of the Early Years. Information Retrieval, 21(2-3), 2018.

Trends in 2024 (1/2)

AGENTFL: Scaling LLM-based Fault Localization to Project-Level Context

Yihao Qin^{*}, Shangwen Wang^{*}, Yiling Lou[†], Jinhao Dong[‡], Kaixin Wang[†], Xiaoling Li^{*}, Xiaoguang Mao^{*},

^{*}National University of Defense Technology, China,

{yihaoqin, wangshangwen13, lixiaoling, xgmao}@nudt.edu.cn

[†]Fudan University, China, {yilinglou@, kxwang23@m.}fudan.edu.cn

[‡]Peking University, China, dongjinhao@stu.pku.edu.cn

Abstract—Fault Localization (FL) is an essential step during the debugging process. With the strong capabilities of code comprehension, the recent Large Language Models (LLMs) have demonstrated promising performance in the code. Nevertheless, due to LLM's limitations in handling long contexts, existing LLM-based methods remain on localizing bugs within a small

scope. This paper leverages Graph Neural Network (GNN) models to learn useful features. Several state-of-the-art techniques have also suggested incorporating auxiliary information, such as code complexity and code history, and features of such information with existing LLM-based models [8], [10].

Test Behavior Tracking,
Document-Guided Search, and
Multi-Round Dialogue

Large Language Models (LLMs) has

Trends in 2024 (2/2)

AGENTLESS 🐱:

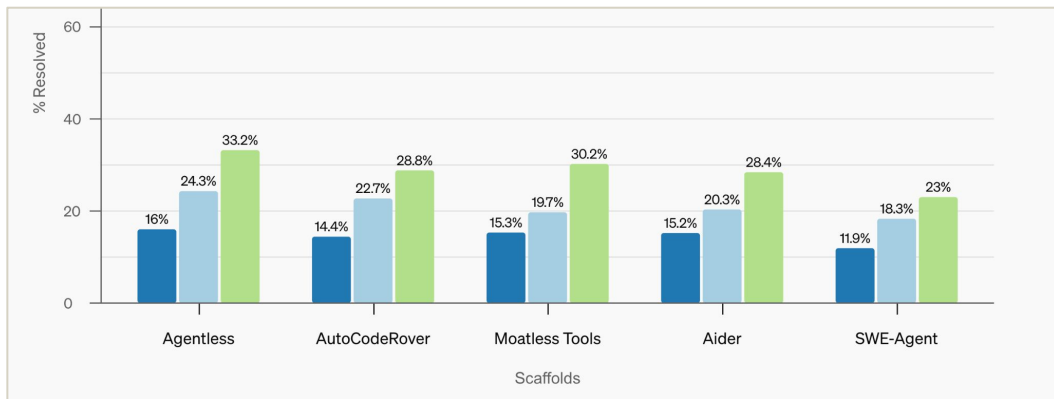
Demystifying LLM-based Software Engineering Agents

Chunqiu Steven Xia* Yinlin Deng* Soren Dunn

Lingming Zhang

University of Illinois Urbana-Champaign 🇺🇸

{chunqiu2, yinlind2, sorend2, lingming}@illinois.edu



Source: <https://openai.com/index/introducing-swe-bench-verified/>

Thank you!

Study data still available today

- <https://mattlease.com/data/ASE2013-BLUIR-Detailed-Results.zip>

ASE 2013 PC led by Tevfik Bultan and Andreas Zeller

ASE 2024 MIP Award Committee led by Myra Cohen and Lars Grunске

Special thanks to Darko Marinov and Lingming Zhang for insightful comments and discussions

Our research was supported in part by NSF Grants SHF-1117902, SRS-0820251, CCF-0845628, and a Temple Fellowship.

Lesson: look beyond your primary area

Take (research-focused) courses in other areas/fields

- Product of Saha taking Lease's "Information Retrieval" class
- Other examples
 - Vasic took Soloveichik's "Unconventional Computing" class, and defined a novel link between traditional CS and synthetic biology [DNA'20 🏆, PNAS'22]
 - Ray took Khurshid's "V&V" class, and defined effective testing of SSL/TLS certificate validation [Oakland'14 🏆]

Collaborate across different research groups

Attend talks in different areas

A few more lessons learned

TA-ing can help with research

- Helping others understand improves the depth of your own understanding
- Basic material from a class you TA can open new research directions

Persevere

- Getting (some) papers rejected is inevitable
- Read the reviews in a positive frame of mind
- Improve the work and re-submit