Automated Bug Management: Reflections & the Road Ahead

SMU
SINGAPORE MANAGEMENT UNIVERSITY

David Lo

School of Computing and Information Systems

Self-Introduction

Self-Introduction

Self-Introduction

Self-Introduction

Singapore Management University

- Third university in Singapore
- Number of students:
  - 9000+ (UG)
  - 2000+ (PG)
- Schools:
  - Computing & IS
  - Economics
  - Law
  - Business
  - Accountancy
  - Social Science

School of Computing and Information Systems

- Undergraduates: 1800+
- Master students: 500+
- Doctoral students: 70+

ARTIFICIAL INTELLIGENCE & DATA SCIENCE

- Data Management & Analytics
- Intelligent Systems & Optimisation
- Machine Learning & Intelligence

HUMAN-MACHINE COLLABORATIVE SYSTEMS

- Pervasive Sensing & Systems
- Multimedia
- Human-Machine Interaction

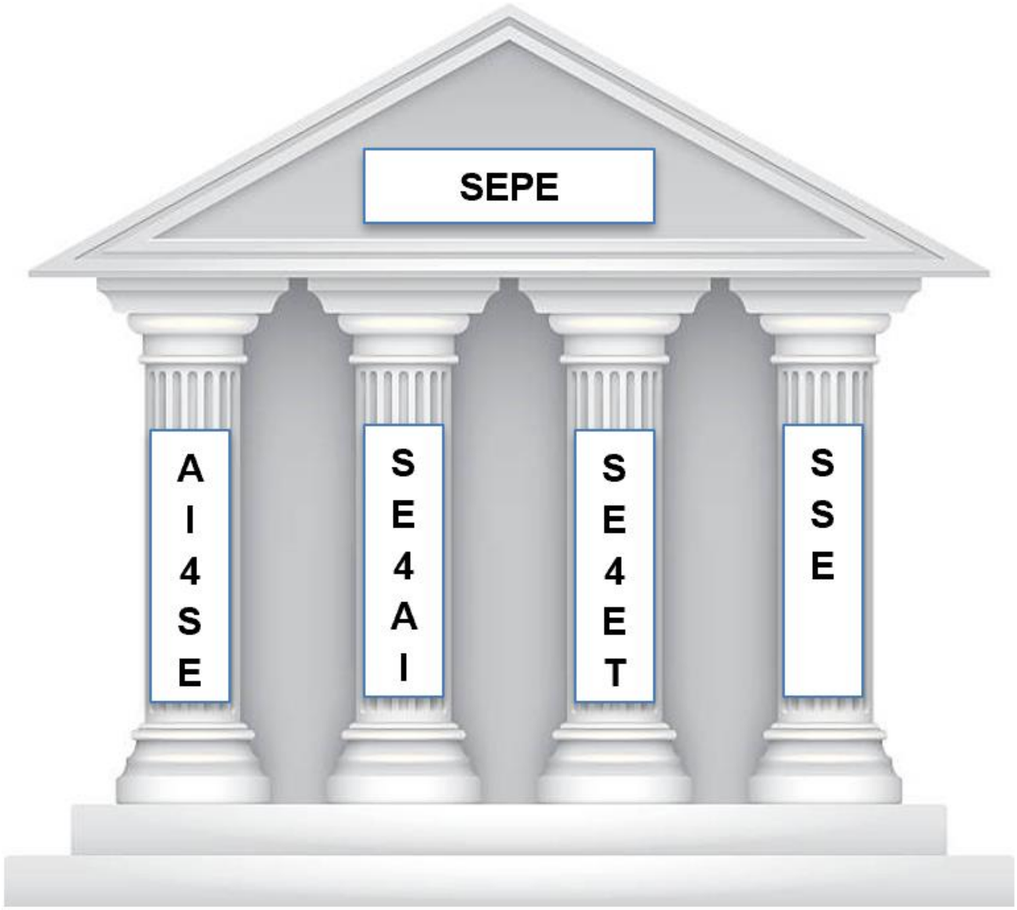
INFORMATION SYSTEMS & TECHNOLOGY

- Software Engineering & Systems
- Cybersecurity
- Information Systems Management

RISE Center (2023-2028)

Center for Research on Intelligent Software Engineering (RISE)

10 faculty members, 40+ staffs and students



AI4SE: AI for SE (*)

SE4AI: SE for AI (*)

SE4ET: SE for Emerging Tech (+)

SSE: Sustainable SE (#)

SEPE: SE Practice Excellence (#)

(*) Established

(+) Growing

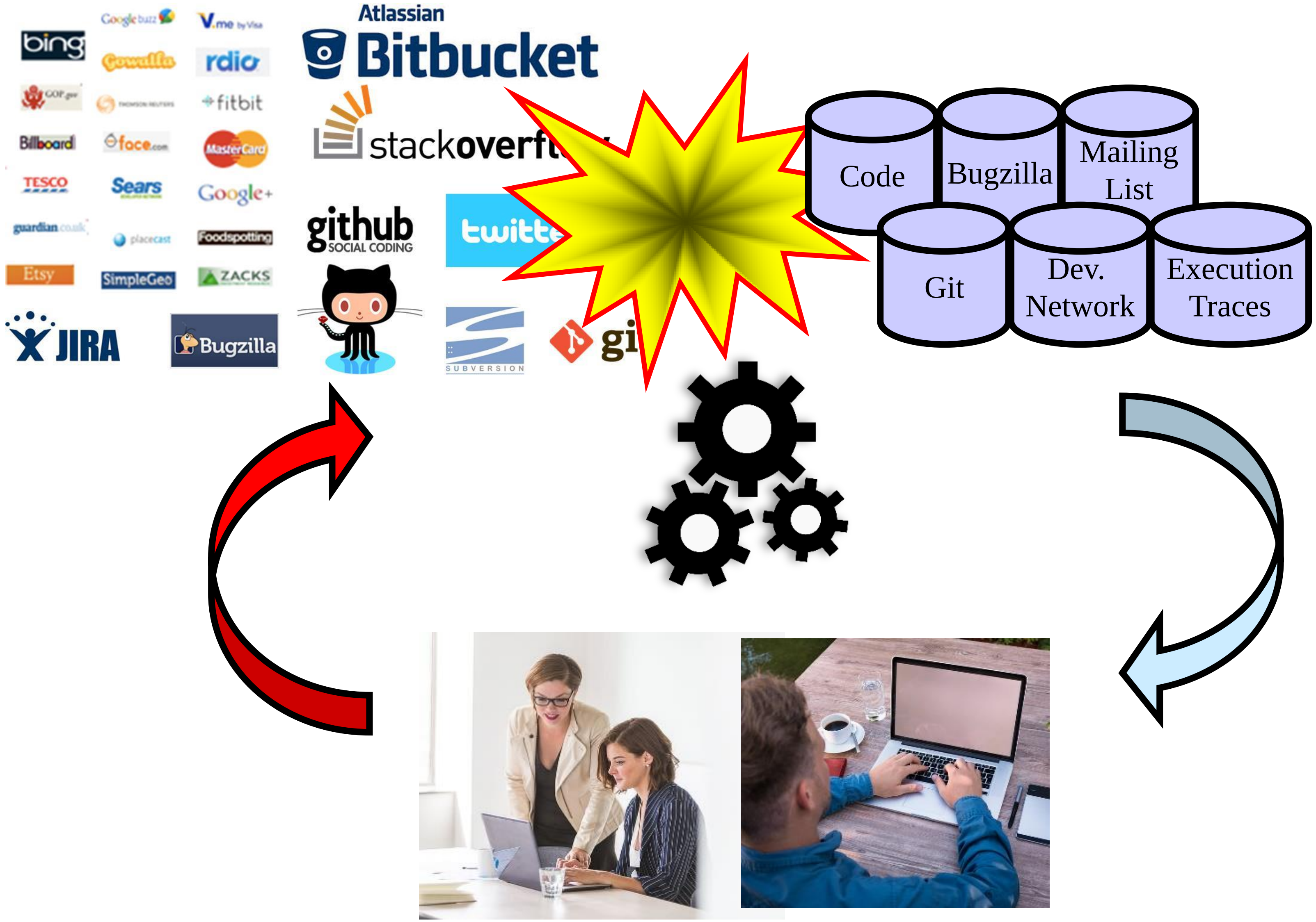
(#) To be developed

SOftware Analytics Research (SOAR) Group

SOAR
SOFTWARE ANALYTICS
RESEARCH GROUP

https://soarsmu.github.io/

SOftware Analytics Research (SOAR) Group



SOftware Analytics Research (SOAR) Group

MONASH University

THE UNIVERSITY OF MELBOURNE

Australian National University

UNSW

CSIRO

DATA 61

Microsoft Research

VERACODE

ZHEJIANG UNIVERSITY 1897

PEKING UNIVERSITY 1898

TEL AVIV UNIVERSITY אוניברסיטת תל אביב

SAP

Tencent 腾讯

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

UNIVERSITEIT STELLENBOSCH UNIVERSITY

SMU SINGAPORE MANAGEMENT UNIVERSITY

UBC

Carnegie Mellon University

IIIT D

inria

SOftware Analytics Research (SOAR) Group

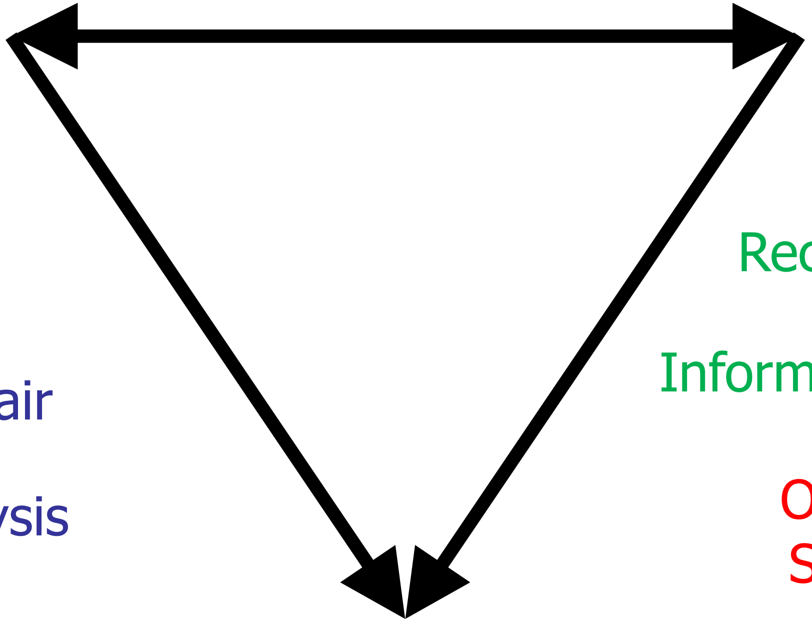
Code Summarization

Smart Contract Analysis

Defect Prediction

Empirical Studies

SE



AI

AI Fairness

AI Testing

AI System Engineering

Question Answering

Pattern / Specification Mining

Automated Test Generation

Bug Finding & Repair

Bug Report Analysis

Recommender Systems

Information Linking

CyberSec

Vulnerability Discovery and Repair

Usable Security

Android Permission and Sandboxing

Open Source Security Risk

Binary Code Analysis

Android Side Channel Analysis

Automated Bug Management: Reflections & the Road Ahead

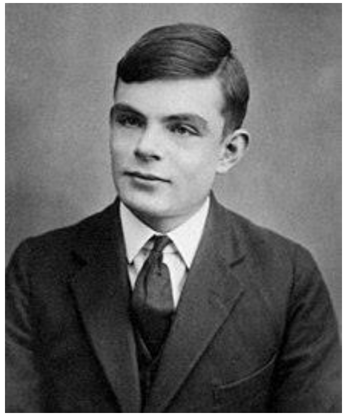
SMU
SINGAPORE MANAGEMENT UNIVERSITY

David Lo

School of
Computing and
Information Systems

Software and Bugs

"There are two ways to write error-free programs; **only the third works**."
-- Alan J. Perlis



"Up to a point, it is better to **just let the snags [bugs] be there** than to spend such time in design that there are none."
-- Alan M. Turing

Bugs and Economy

- Software bugs are prevalent
- Although most bugs are less “harmful”, many have serious impact
  - Collectively responsible for trillions of dollars

FEATURE

Study: Buggy software costs users, vendors nearly $60B annually

By Patrick Thibodeau

Senior Editor, Computerworld | JUN 25, 2002 12:00 AM PST

CISQ

Consortium for Information & Software Quality ™

Our 2022 update report estimates that the cost of poor software quality in the US has grown to at least $2.41 trillion, but not in similar proportions as seen in 2020. The accumulated software Technical Debt (TD) has grown to ~$1.52 trillion.

Bug Management and Need for Automation

- Developers often receive more bugs than they can handle
- Developers spend a lot of time debugging

Survey: Fixing Bugs Stealing Time from Development

BY: MIKE VIZARD ON FEBRUARY 16, 2021 — 4 COMMENTS

A global survey of 950 developers published today finds more than a third (38%) of developers spend up to a quarter of their time fixing software bugs, with slightly more than a quarter (26%) spending up to half their time fixing bugs.

The `source` property of problem matchers is not documented
#182168 opened 22 minutes ago by akbyrd

Editor languate status busy icon, most distracting thing ever!!
#182166 opened 41 minutes ago by emaayan

Python multi-line comment strings should be treated as comments in the Monokai theme
#182163 opened 1 hour ago by ElectricRCAircraftGuy

Abnormal information was generated when testing vscode!
#182161 opened 1 hour ago by DigOrDog

Keybinding for `git commit -m`
#182158 opened 1 hour ago by User087

Bug Management and Need for Automation

Coping with an Open Bug Repository

John Anvik, Lyndon Hiew and Gail C. Murphy
Department of Computer Science
University of British Columbia
{janvik, lyndonh, murphy}@cs.ubc.ca

ETX 2005

*"Everyday, almost **300 bugs** appear that need triaging. This is far too much for only the Mozilla programmers to handle." – Mozilla Developer*

Bug Management and Need for Automation

Emerging App Issue Identification from User Feedback: Experience on WeChat

Cuiyun Gao†, Wujie Zheng§*, Yuetang Deng§, David Lo‡, Jichuan Zeng†, Michael R. Lyu†, Irwin King†
†Dept. of Computer Science and Engineering, The Chinese University of Hong Kong, China
‡School of Information Systems, Singapore Management University, Singapore
§Tencent, Inc., China
{wujiezheng,yuetangdeng}@tencent.com, {cygao,jczeng,lyu,king}@cse.cuhk.edu.hk, davidlo@smu.edu.sg

ICSE 2019

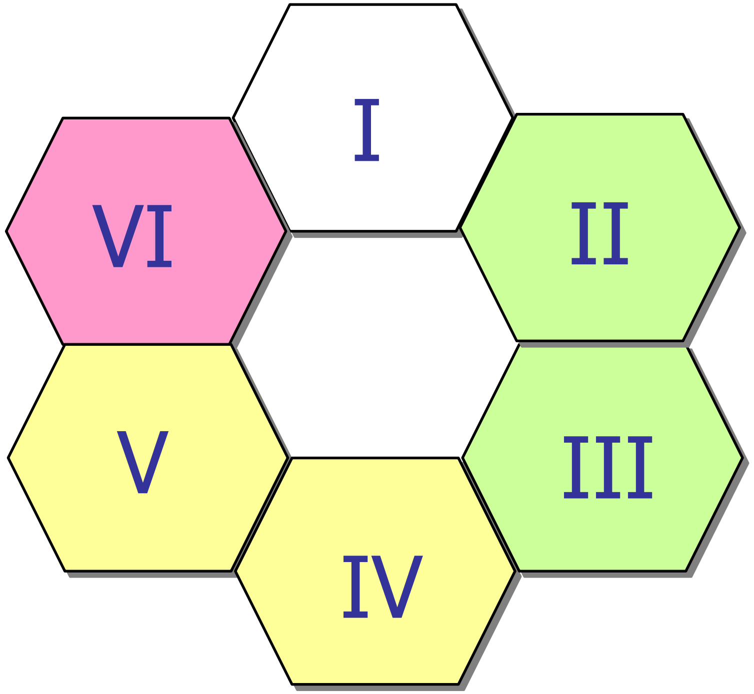
*"Unfortunately, the large quantity of user reviews (e.g., WeChat receives around **60,000 reviews** per day) makes manual analysis inefficient and unrealistic."*

Key Question

Can we help developers by **automating** some tasks in the bug report management process?

Outline

Motivation



Open Challenges

Data

Duplicate Detection

Tasks

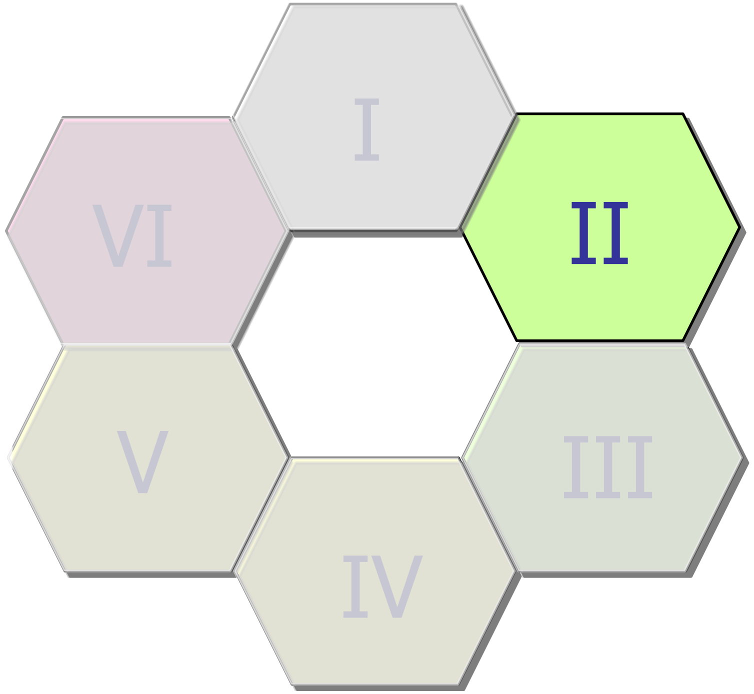
Bug Localization

SMU SINGAPORE MANAGEMENT UNIVERSITY

School of Computing and Information Systems

Outline

Motivation



Data

Tasks

Bug Localization

Duplicate Detection

Open Challenges

I. What Data Can We Mine?

- People report errors and incidents that they encounter when using a software
- These reports often include:
  - Description of the issue
  - Steps to reproduce the issue
  - Severity level
  - Parts of the system affected by the issue
  - Failure traces
- Come in various “shapes and sizes”

Issue Report – Bugzilla, Manually Submitted

On First opening Firefox, the software hangs for 10 to 20 seconds when anthing is entered into the address bar

Title

▾ Categories

Product: Firefox ▾
Component: Address Bar ▾
Version: 68 Branch

Type: defect

Priority: P3 Severity: S3
Points: 5

Informative Fields

▾ Tracking

Status: UNCONFIRMED

Issue Report – Bugzilla, Manually Submitted

User Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:68.0) Gecko/20100101 Firefox/68.0

Steps to reproduce:

We have Firefox 68.5.0ESR. Installed using the .msi file via SCCM. We have a proxy server. GPO sets the following:
We can then set the following policies:
• Set the ImportEnterpriseRoots key to true.
• Under Proxy, set "Mode" to "autoDetect"
• DisableAppUpdate to true
• Under Homepage, setting URL to [intranet page], and StartPage to "homepage"
• Under PopupBlocking, setting "allow" to [intranet page];
• Under FlashPlugin, setting "allow" to [intranet page]
• Under OverrideFirstRunPage, set to Blank, so it will default to the homepage
• Under OverridePostUpdatePage, set to Blank, so it will default to the homepage

Detailed Description

Issue Report – JIRA, Manually Submitted

NetBeans / NETBEANS-5735

Fatal exception starting Glassfish 6.1

Title

Informative Fields

Detailed Description

Details

Type:	Bug
Status:	OPEN
Priority:	Blocker
Resolution:	Unresolved

Description

When I try to start Glassfish-6.1 within the Services-Server-Tab with JDK-1.8 it does not start, instead the following is displayed:

Error: Could not create the Java Virtual Machine.

Issue Report – Bugzilla, Submitted by Bot

Update Bot Assignee
Comment 1 • 1 month ago

I've submitted a try run for this commit: https://treeherder.mozilla.org/#/job

Update Bot Assignee
Comment 2 • 1 month ago

The try push is done, we found jobs with unclassified failures.

Known Issues (From Push Health):

```
browser/components/extensions/test/browser/browser_ext_
      - 4 of 4 failed on the same (retriggered) task
```

CI Failures

Issue Report – Google Play Review

Rating

Wesley

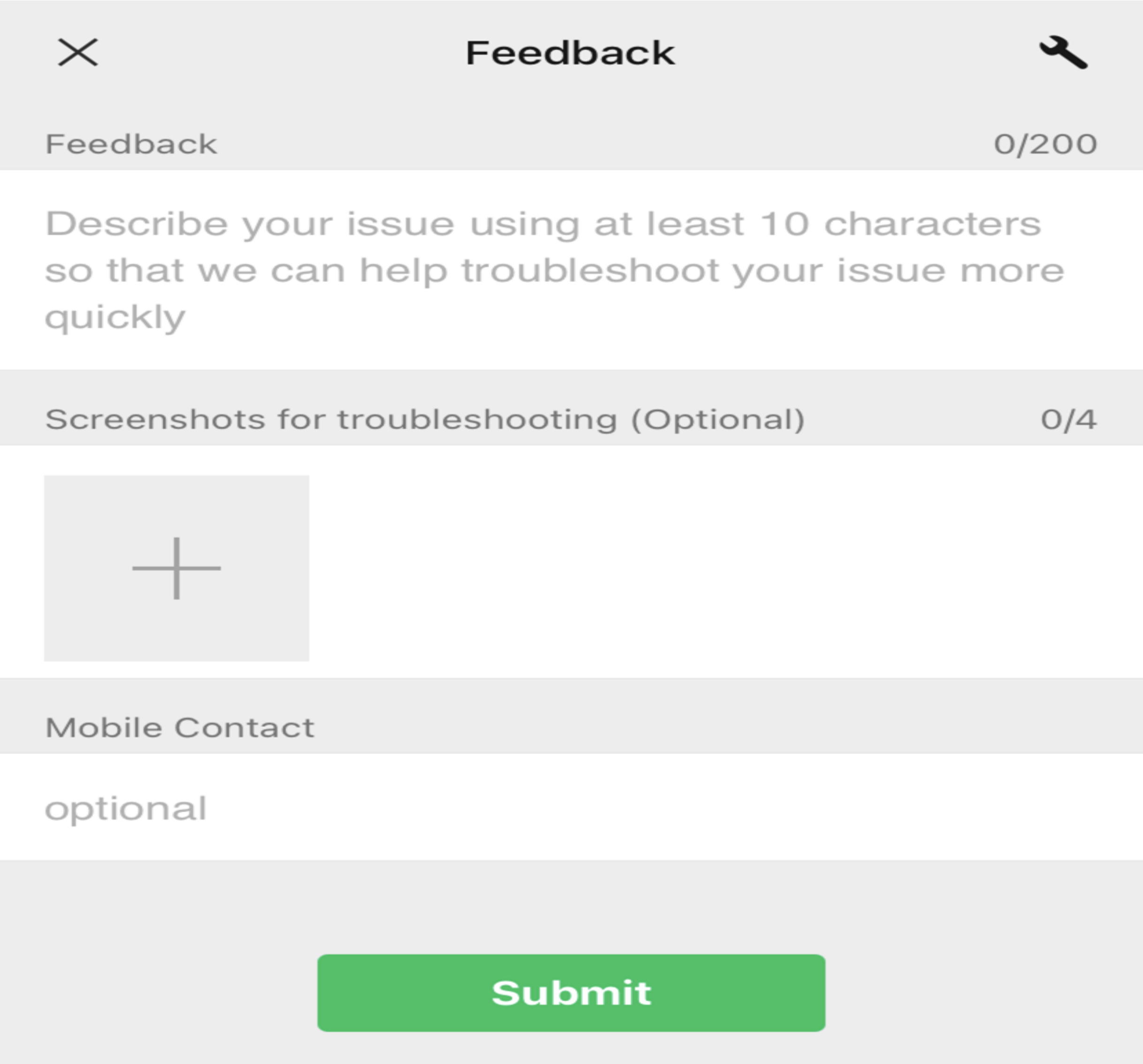
★★★☆☆ July 2, 2021

1387

The muted feedback option is bugged I'm sure. Before I could preview videos without sound and I was ok with that. But now there's an option to either have it on with sound playing when you look at the preview which at that point you might as well just click the video or turn it off but it turns off ...

Comment

Issue Report – WeChat Feedback Form



Commits Linked to Issue Reports

Camel / CAMEL-16574

camel-salesforce - NPE On Response Callback - DefaultCompositeApiClient

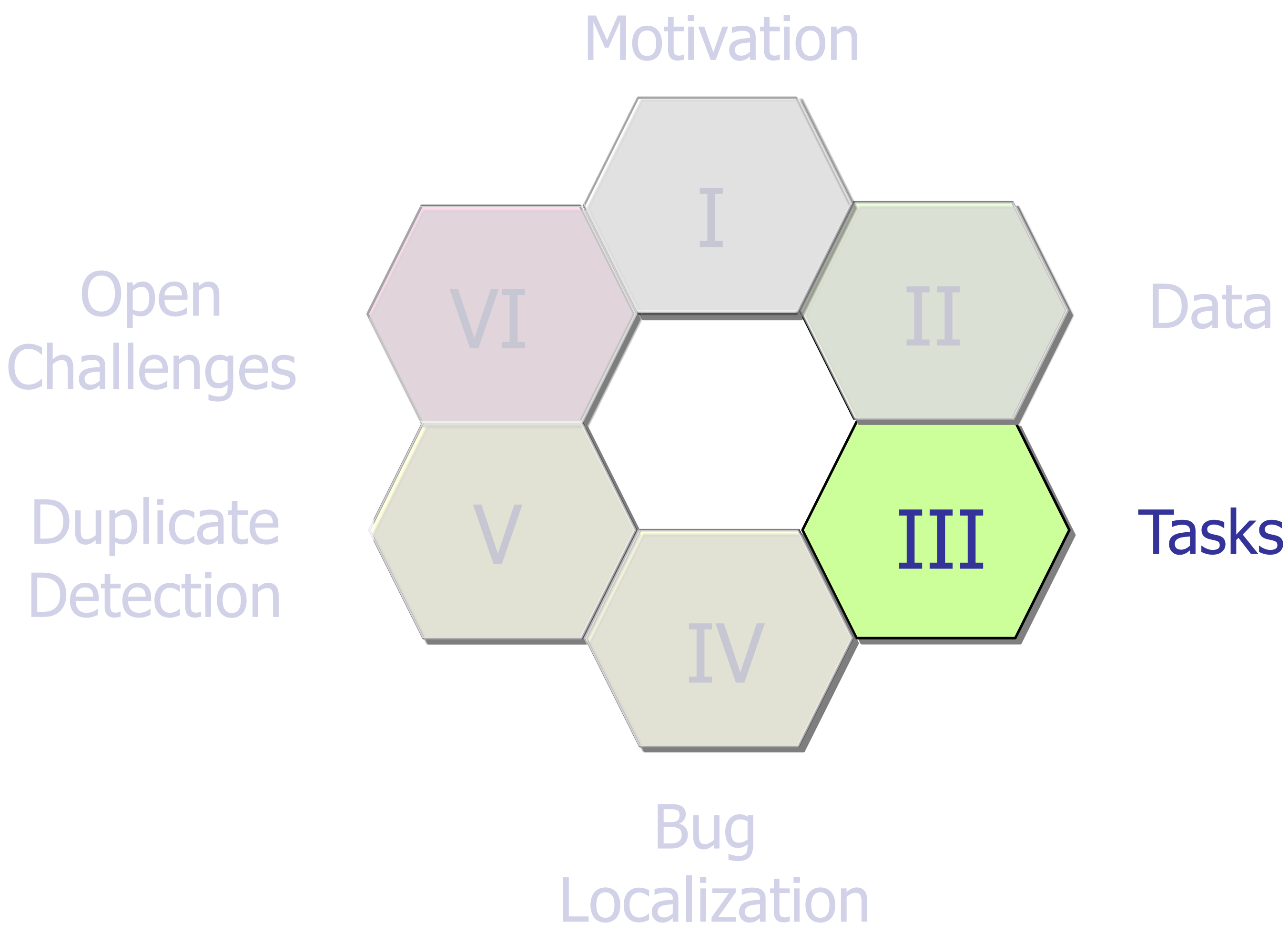
✓ **CAMEL-16574: Fixed NPE Exception On Empty Response (#5502)**

main (#5502)

camel-3.11.0 camel-3.10.0

jawad-r3 committed on 5 May Verified

Outline


Motivation
I
Data
II
Tasks
III
Bug Localization
IV
Duplicate Detection
V
Open Challenges
VI

II. What Tasks Can We Automate?

How Practitioners Perceive Automated Bug Report Management Techniques

Weiqin Zou, David Lo, Zhenyu Chen, Xin Xia, Yang Feng, Baowen Xu

Abstract—Bug reports play an important role in the process of debugging and fixing bugs. To reduce the burden of bug report managers and facilitate the process of bug fixing, a great amount of software engineering research has been invested toward automated bug report management techniques. However, the verdict is still open whether such techniques are actually required and applicable outside the domain of theoretical research. To fill this gap, we conducted a survey among 327 practitioners to gain their insights into various categories of automated bug report management techniques. Specifically, we asked the respondents to rate the importance of such techniques and provide the rationale. To get deeper insights into practitioners' perspective, we conducted follow-up interviews with 25 interviewees selected from the survey respondents. Through the survey and the interviews, we gained a better understanding of the perceived usefulness (or its lack) of different categories of automated bug report management techniques. Based on our findings, we summarized some potential research directions in developing techniques to help developers better manage bug reports.

TSE 2020

Literature Review

- **Paper Selection**
  - 7 journals and 10 conferences
  - Years (2006-2017)
  - Regular Papers
  - Non-Empirical Studies
  - Card Sorting

115 papers of 10 categories

Journals

TOSEM, TSE, EMSE, ASEJ, JSS, IST, TRel

Conferences

ICSE, FSE, ASE, ICSME, ICPC, ISSTA, SANER, ESEM, ICST, MSR

Literature Review

ID	Category	Total
T1	Bug localization	24
T2	Bug assignment	22
T3	Duplicate/similar bug detection	14
T4	Bug categorization	12
T5	Bug fixing time prediction	10
T6	Bug severity/priority prediction	8
T7	Bug report completion/refinement	8
T8	Bug-commit linking	7
T9	Bug report summarization/visualization	5
T10	Reopened bug prediction	5

List of papers:

Automatable Tasks (T1)

Bug Localization

These techniques process a bug report, and **locate relevant program elements** that possibly **contain the bug**. Some of these techniques also recommend candidate **repairs**.

Where Should the Bugs Be Fixed?

More Accurate Information Retrieval-Based Bug Localization Based on Bug Reports

Jian Zhou[1], Hongyu Zhang [1,*] and David Lo[2]
[1]*School of Software, Tsinghua University, Beijing 100084, China*
Tsinghua National Laboratory for Information Science and Technology (TNList)
zhoujian1286@yahoo.com.cn, hongyu@tsinghua.edu.cn
[2]*School of Information Systems, Singapore Management University, Singapore*
davidlo@smu.edu.sg

ICSE 2012

Automatable Tasks (T2)

Bug Assignment

These techniques process a bug report, and recommend the most **appropriate developers to address it**.

Who Should Fix This Bug?

ICSE 2006
MIP ICSE 2016

John Anvik, Lyndon Hiew and Gail C. Murphy
Department of Computer Science
University of British Columbia
{janvik, lyndonh, murphy}@cs.ubc.ca

ABSTRACT

Open source development projects typically support an open bug repository to which both developers and users can re-

However, this potential advantage also comes with a significant cost. Each bug that is reported must be *triaged* to determine if it describes a meaningful new problem or

Automatable Tasks (T3)

Duplicate / Similar Bug Detection

These techniques detect **duplicate / similar reports** in issue tracking systems.

A Discriminative Model Approach for Accurate Duplicate Bug Report Retrieval

ICSE 2010

Chengnian Sun[1], David Lo[2], Xiaoyin Wang[3], Jing Jiang[2], Siau-Cheng Khoo[1]
[1]School of Computing, National University of Singapore
[2]School of Information Systems, Singapore Management University
[3]Key laboratory of High Confidence Software Technologies (Peking University), Ministry of Education
suncn@comp.nus.edu.sg, davidlo@smu.edu.sg, wangxy06@sei.pku.edu.cn,
jingjiang@smu.edu.sg, khoosc@comp.nus.edu.sg

ABSTRACT

Bug repositories are usually maintained in software projects. Testers or users submit bug reports to identify various issues with systems. Sometimes two or more bug reports correspond to the same defect. To address the problem with duplicate bug reports, a person called a triager needs to manually label these bug reports as duplicates, and link them

Due to complexities of systems built, software often comes with defects. Software defects have caused billions of dollars lost [20]. Fixing defects is one of the most frequent reasons for software maintenance activities which also goes to 70 billion US dollars in the United States alone [19].

In order to help track software defects and build more reliable systems, bug tracking tools have been introduced. Bug

SMU SINGAPORE MANAGEMENT UNIVERSITY

School of Computing and Information Systems

Automatable Tasks (T4)

Bug Categorization

These techniques process a bug report, and classify it into **different categories** (e.g. invalid or not, bug or feature request, security bug report or not etc.)

Automatic Defect Categorization

WCRE 2012

Ferdian Thung, David Lo, and Lingxiao Jiang
School of Information Systems
Singapore Management University, Singapore
{ferdianthung,davidlo,lxjiang}@smu.edu.sg

Abstract—Defects are prevalent in software systems. In order to understand defects better, industry practitioners often categorize bugs into various types. One common kind of categorization is the IBM's Orthogonal Defect Classification (ODC). ODC proposes various orthogonal classification of defects based on much information about the defects, such as the symptoms and semantics of the defects, the root cause analysis of the defects, and many more. With these category labels, developers can better

In this paper, we propose a classification-based approach that categorizes defects into three families: control and data flow, structural, and non-functional. Our goal is to automatically classify a defect into one of the three families according to the content of the bug report and the associated code changes made to fix the bug. Given a large set of data about known defects and their fixes, various textual features corresponding

SMU SINGAPORE MANAGEMENT UNIVERSITY

School of Computing and Information Systems

Automatable Tasks (T5)

Bug Fixing Time Prediction

These techniques process a bug report, and predict **how long it will take to fix the bug**.

Bug-fix Time Prediction Models: Can We Do Better?

Pamela Bhattacharya Iulian Neamtiu
Department of Computer Science and Engineering
University of California, Riverside, CA, USA
{pamelab,neamtiu}@cs.ucr.edu

MSR 2011

ABSTRACT

Predicting bug-fix time is useful in several areas of software evolution, such as predicting software quality or coordinating development effort during bug triaging. Prior work has proposed bug-fix time prediction models that use various bug report attributes (e.g., number of developers who participated in fixing the bug, bug severity, number of patches, bug-opener's reputation) for estimating the time it will take to fix a newly-reported bug. In this paper we take a

1. INTRODUCTION

Predicting bug-fix time is useful in several areas of software evolution, such as predicting software quality [9] or coordinating effort during bug triaging [8]. To this end, prior efforts have constructed bug-fix time prediction models, based on machine learning algorithms, [1] on both open source and commercial projects.

Prior studies on open source projects [8, 5, 1] have used various bug report attributes (e.g., number of developers involved in fixing

Automatable Tasks (T6)

Bug Severity / Priority Prediction

These techniques process a bug report, and predict its **severity / priority**.

Automated Severity Assessment of Software Defect Reports

ICSM 2008

Tim Menzies
Lane Department of Computer Science,
West Virginia University
PO Box 6109, Morgantown, WV, 26506
304 293 0405
tim@menzies.us

Andrian Marcus
Department of Computer Science
Wayne State University
Detroit, MI 48202
313 577 5408
amarcus@wayne.edu

Abstract

In mission critical systems, such as those developed by NASA, it is very important that the test engineers properly recognize the severity of each issue they identify during testing. Proper severity assessment is essential for appropriate resource allocation and planning for fixing activities and additional testing.

of the projects collected issue data. In most instances, the specific configuration of the information captured about an issue was tailored by the IV&V project to meet its needs. This has created consistency problems when metrics data is pulled across projects. While there was a set of required data fields, the majorities of those fields do not provide information in regards to

Automatable Tasks (T7)

Bug Report Completion / Refinement

These techniques aim to generate a high-quality bug report. Some of these techniques automatically **generate a new bug report when software crashes**. Some others **enrich / modify** an existing one.

Auto-completing Bug Reports for Android Applications

FSE 2015

Kevin Moran, Mario Linares-Vásquez, Carlos Bernal-Cárdenas, Denys Poshyvanyk
College of William & Mary
Department of Computer Science
Williamsburg, VA 23187-8795, USA
{kpmoran, mlinarev, cebernal, denys}@cs.wm.edu

ABSTRACT

The modern software development landscape has seen a shift in focus toward mobile applications as tablets and smartphones near ubiquitous adoption. Due to this trend, the complexity of these "apps" has been increasing, making development and maintenance challenging. Additionally, current bug tracking systems are not able to effectively support construction of reports with actionable information that directly lead to a bug's resolution. To address the need for an improved reporting system, we introduce a novel solution,

1. INTRODUCTION

Smartphones and mobile computing have skyrocketed in popularity in recent years, and adoption has reached near-ubiquitous levels with over 2.7 billion active smartphone users in 2014 [36]. An increased demand for high-quality, robust mobile applications is being driven by a growing user base that performs an increasing number of computing tasks on "smart" devices. Due to this demand, the complexity of mobile applications has been increasing, making development and maintenance challenging. The intense competition

SMU SINGAPORE MANAGEMENT UNIVERSITY

School of Computing and Information Systems

Automatable Tasks (T8)

Bug-Commit Linking

These techniques aim to **link bug reports with bug fixing commits or bug inducing commits**. With these techniques, developers can better understand which commits fix the bug and why/how/when the bug is introduced.

RCLinker: Automated Linking of Issue Reports and Commits Leveraging Rich Contextual Information

ICPC 2015

Tien-Duy B. Le*, Mario Linares-Vásquez†, David Lo*, and Denys Poshyvanyk†
*School of Information Systems, Singapore Management University
†Computer Science Department, The College of William and Mary
{btdle.2012,davidlo}@smu.edu.sg, {mlinarev,denys}@cs.wm.edu

***Abstract*—Links between issue reports and their corresponding commits in version control systems are often missing. However, these links are important for measuring the quality of a software system, predicting defects, and many other tasks. Several approaches have been designed to solve this problem by automatically linking bug reports to source code commits via comparison of textual information in commit messages and bug reports. Yet, the effectiveness of these techniques is**

approaches named *ReLink* [60] and *MLink* [47] respectively. These approaches enumerate a set of potential links and remove the ones that do not satisfy some criteria defined based on a set of thresholds. These thresholds are learned by heuristically enumerating various values based on a training and/or a validation dataset. A main operation in *ReLink* and *MLink* is the computation of similarity between the textual

Automatable Tasks (T9)

Bug Report Summarization / Visualization

These techniques process a bug report, and summarize it into a **much shorter form**. Some of these techniques also help developers better navigate / understand bug reports through visualization.

Automatic Summarization of Bug Reports

TSE 2014

Sarah Rastkar, Gail C. Murphy, *Member, IEEE*, and Gabriel Murray

Abstract—Software developers access bug reports in a project's bug repository to help with a number of different tasks, including understanding how previous changes have been made and understanding multiple aspects of particular defects. A developer's interaction with existing bug reports often requires perusing a substantial amount of text. In this article, we investigate whether it is possible to summarize bug reports automatically so that developers can perform their tasks by consulting shorter summaries instead of entire bug reports. We investigated whether existing conversation-based automated summarizers are applicable to bug reports and found that the quality of generated summaries is similar to summaries produced for e-mail threads and other conversations. We also trained a summarizer on a bug report corpus. This summarizer produces summaries that are statistically better than summaries produced by existing conversation-based generators. To determine if automatically produced bug report summaries can help a developer with their work, we conducted a task-based evaluation that considered the use of summaries for bug report duplicate detection tasks. We found that summaries helped the study participants save time, that there was no evidence that accuracy degraded when summaries were used and that most participants preferred working with summaries to working with original bug reports.

Automatable Tasks (T10)

Reopened Bug Prediction

These techniques process a closed bug report, and predict whether it is likely to be **reopened**.

Characterizing and Predicting Which Bugs Get Reopened

Thomas Zimmermann [1] tzimmer@microsoft.com
Nachiappan Nagappan [1] nachin@microsoft.com
Philip J. Guo [2] pg@cs.stanford.edu
Brendan Murphy [3] bmurphy@microsoft.com
[1] Microsoft Research, USA [2] Stanford University, USA [3] Microsoft Research, UK

Abstract—**Fixing bugs is an important part of the software development process. An underlying aspect is the effectiveness of fixes: if a fair number of fixed bugs are reopened, it could indicate instability in the software system. To the best of our knowledge there has been on little prior work on understanding the dynamics of bug reopens. Towards that end, in this**

the likelihood of being reopened, but to characterize the overall reopen process. In order to do so we employ a more foundational approach wherein we first survey a large population of experienced developers on the fundamental reasons for bug reopens and qualitatively analyze the responses. We then assess the reasons for reopens from a quantitative per-

ICSE 2012

Practitioners’ Perceptions

Potential gap between research and practice in automated bug report management:

- Are these techniques appreciated by practitioners?
- What are practitioners’ complaints and challenges?

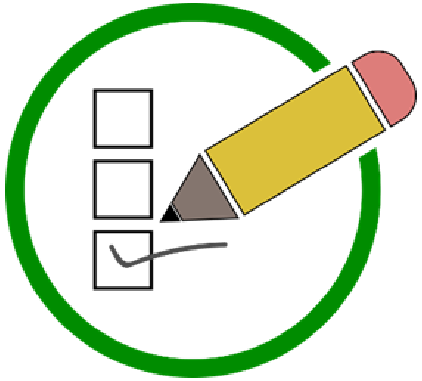
This study:

- Investigates practitioner views of existing research work
- Identifies new research directions by learning from practice

Methodology

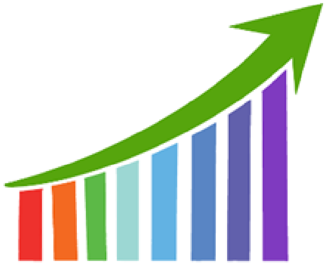
① **Literature Review**

Categories



② **Survey**

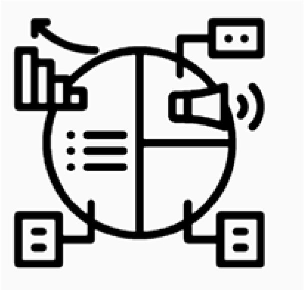
Ratings + Comments



Statistical Analysis

③ **Interview**

Comments



Thematic Analysis

④ **Data Analysis**

Surveys

Questionnaire

- Demographic questions
- 10 closed-ended questions for T1-T10 (very important->very unimportant)
- Up to 2 open-ended questions asking rationales of important/unimportant ratings

Participants

327 respondents

Industrial professionals

Open-source developers

Interviews

Procedure

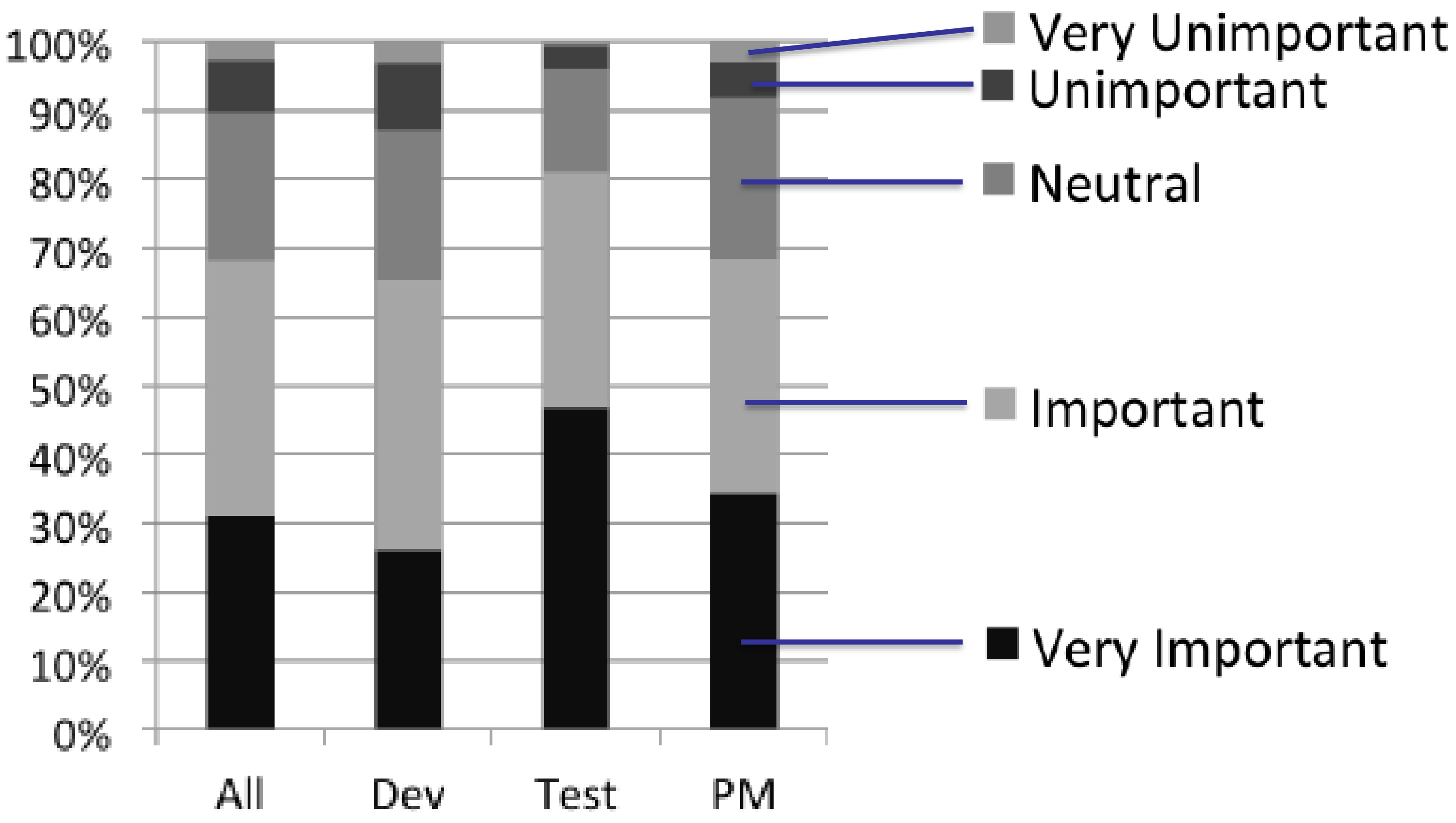
~1 hour, using video conferencing or in-person

4 out of 10 categories with each interviewee

Participants

25 out of 107 survey respondents who left email addresses in the anonymous survey

Findings



~70% of ratings (**>80%** for testers)
are important / very important

Findings

ID	Technique	5 (Very Important)	4	3	2	1 (Very Unimportant)
T1	Bug localization	**49.2%**	33.4%	9.6%	6.2%	1.5%
T2	Bug assignment	33.4%	**38.9%**	19.0%	5.8%	2.8%
T3	Duplicate / similar bug detection	35.8%	**41.6%**	18.3%	3.4%	0.9%
T4	Bug categorization	33.3%	**41.0%**	17.7%	5.5%	2.4%
T5	Bug fixing time prediction	17.8%	28.3%	**31.2%**	15.3%	7.5%

Findings

		Very Important				Very Unimportant
ID	**Technique**	**5**	**4**	**3**	**2**	**1**
T6	Bug severity / priority prediction	25.6%	**37.3%**	26.9%	8.3%	1.9%
T7	Bug report completion / refinement	34.5%	**42.8%**	18.8%	3.1%	0.9%
T8	Bug-commit linking	36.7%	**44.4%**	15.1%	2.8%	0.9%
T9	Bug report summarization / visualization	25.9%	**34.9%**	25.6%	10.8%	2.8%
T10	Reopened bug prediction	17.8%	29.1%	**32.2%**	15.0%	5.9%

Findings

Ranks of ten categories of automated bug report management techniques according to the Scott-Knott ESD tests for all respondents.

Group	Category of Bug Report Management Technique
1	T1 (Bug localization) T3 (Duplicate/similar bug detection) T7 (Bug report completion/refinement) T8 (Bug-commit linking)
2	T2 (Bug assignment) T4 (Bug categorization)
3	T9 (Bug report summarization/visualization) T6 (Bug severity/priority prediction)
4	T5 (Bug fixing time prediction) T10 (Re-opened bug prediction)

Highest Rank

Lowest Rank

Summary of **Green** Segment (Parts II, III)

- Bug reports come in various shapes and sizes
- Hundreds of papers on AI for issue management
  - Since "Who Should Fix this Bug?" (MIP ICSE 2016)
  - Categorized into 10 categories
- Perceived as important by practitioners (70-80%); top-4:
  - Bug Localization
  - Duplicate / Similar Bug Report Detection
  - Bug Report Completion / Refinement
  - Bug-Commit Linking

Data

II

Tasks

III

SINGAPORE MANAGEMENT UNIVERSITY

School of Computing and Information Systems

Outline

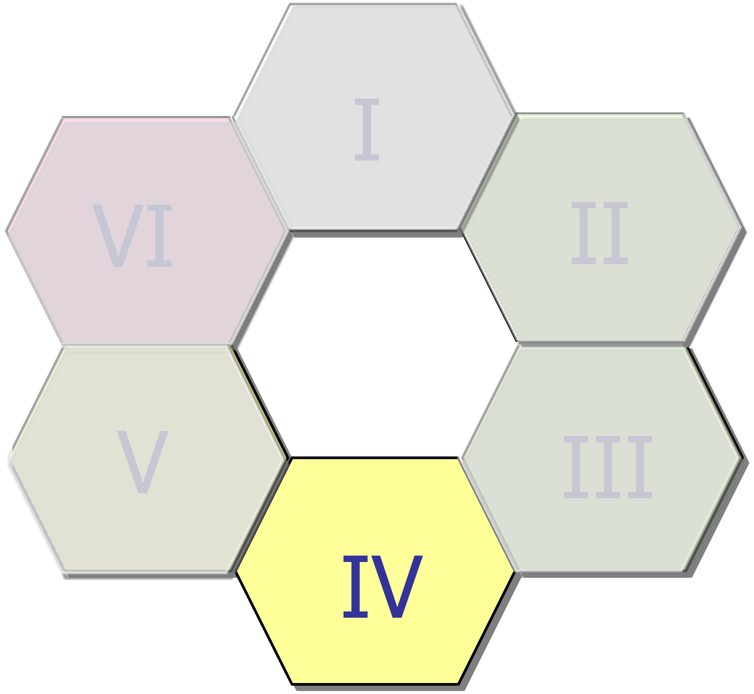
Motivation

Open Challenges

Data

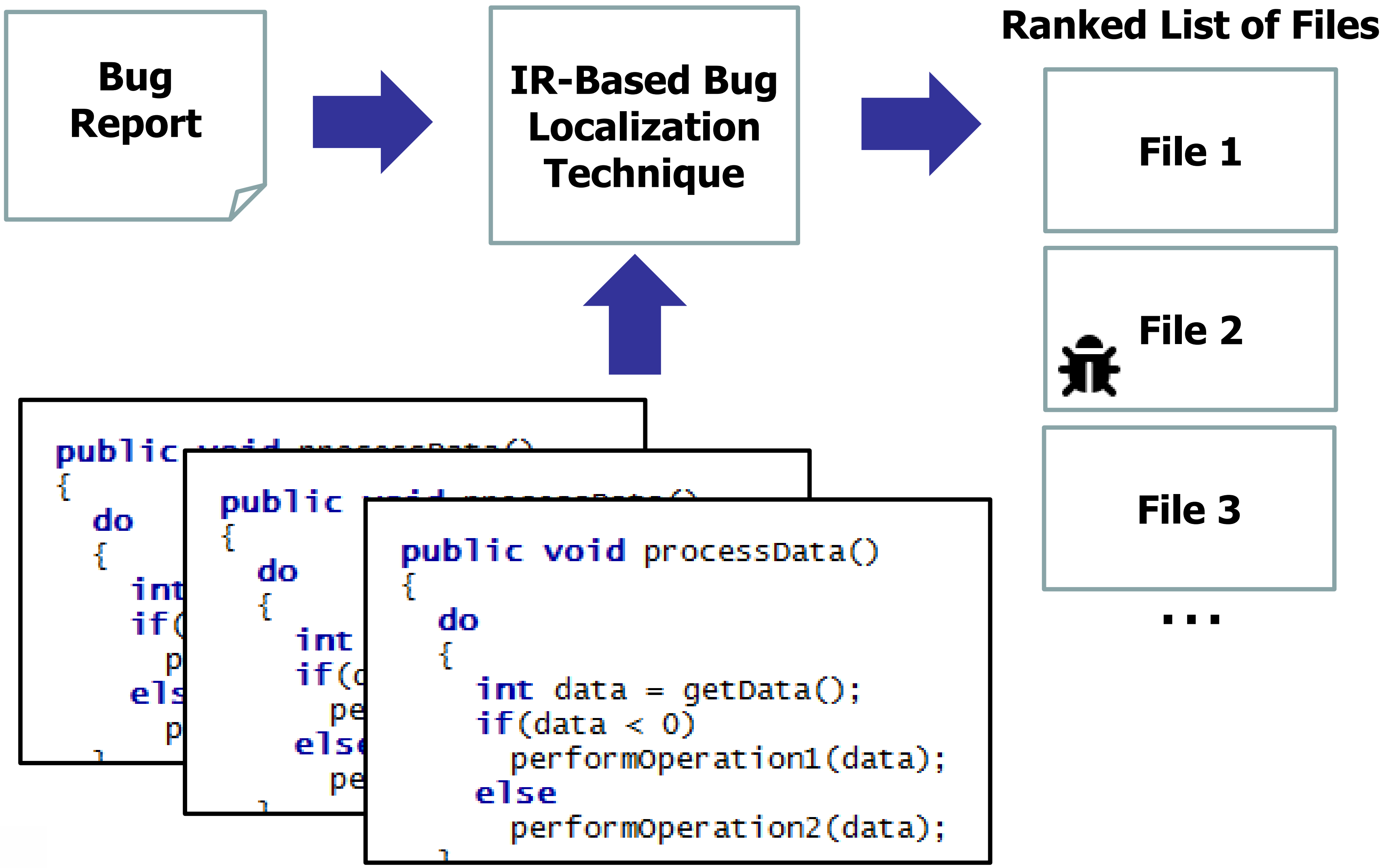
Duplicate Detection

Tasks



Bug Localization

IV. Bug Localization



(Thousands of) Source Code Files

Popular Early Work

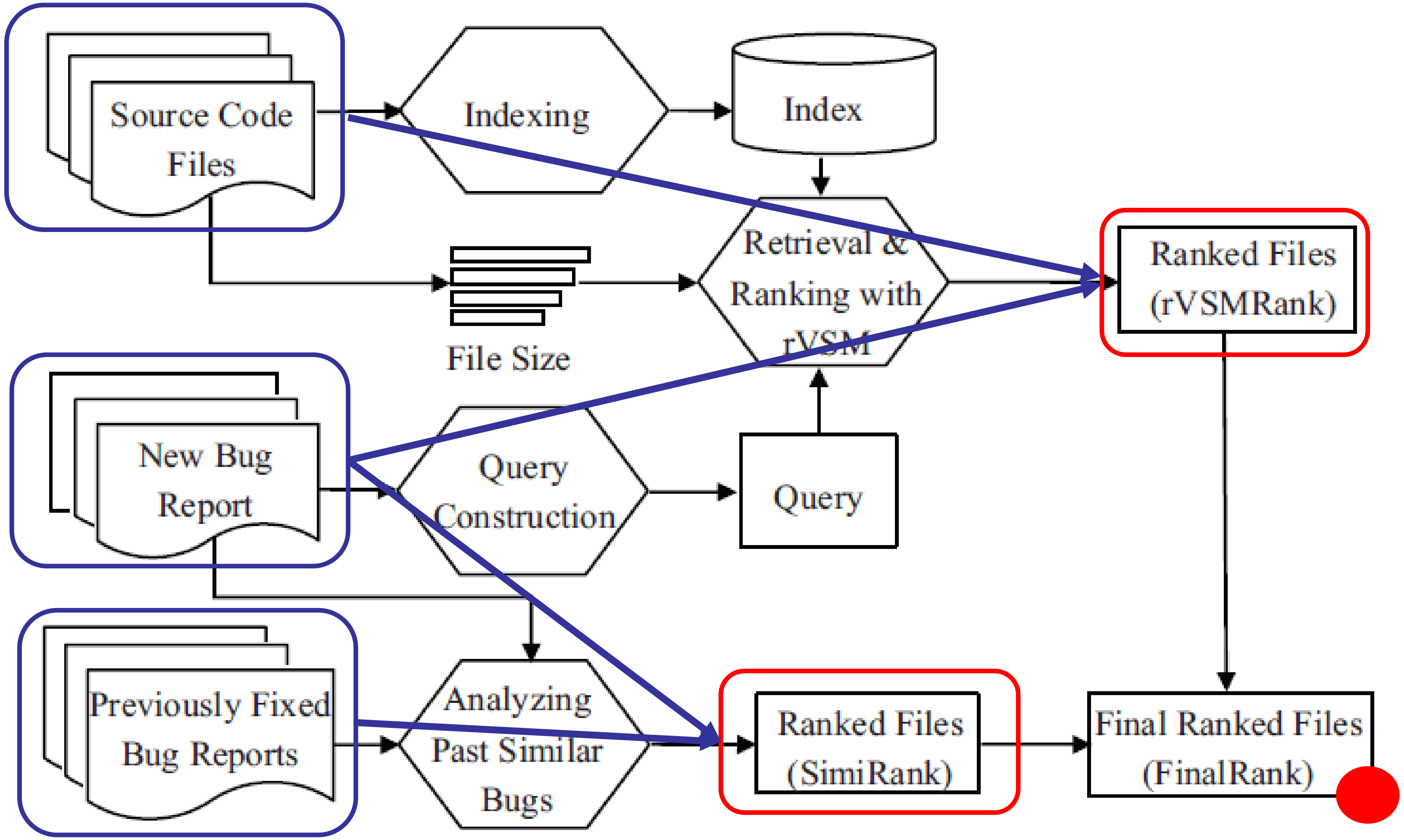
Where Should the Bugs Be Fixed?

More Accurate Information Retrieval-Based Bug Localization Based on Bug Reports

Jian Zhou[1], Hongyu Zhang [1,*] and David Lo[2]

ICSE 2012

BugLocator



$$FinalScore = (1-\alpha)\times N(rVSMScore) + \alpha \times N(SimiScore)$$

rVSM Score

$$rVSMScore(q,d) = g(\#term) \times \cos(q,d)$$

$$\cos(q,d) = \frac{\vec{V_q} \bullet \vec{V_d}}{|\vec{V_q}||\vec{V_d}|}$$

Classical VSM with

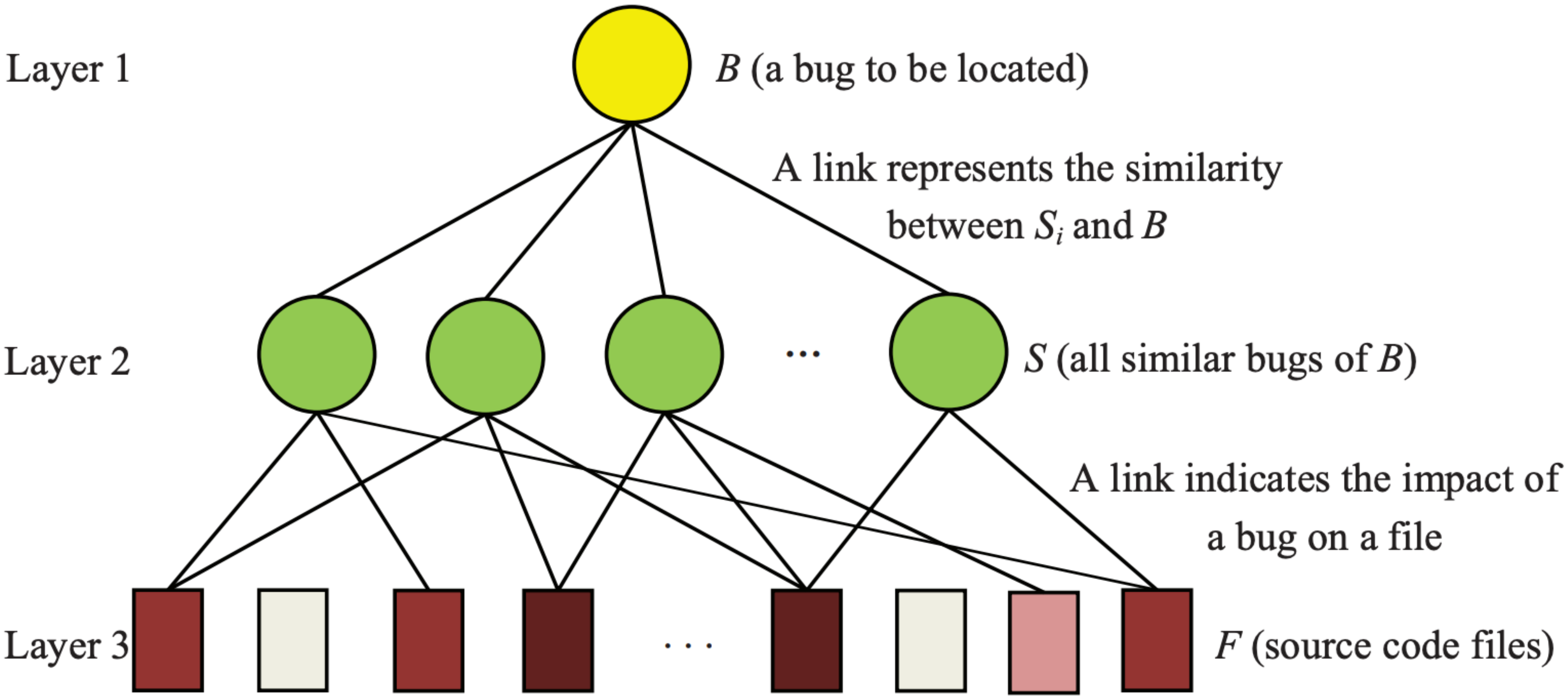
$$tf(t,d) = \log(f_{td}) + 1$$

$$idf(t) = \log(\frac{\#docs}{n_t})$$

$$g(\#terms) = \frac{1}{1+e^{-N(\#terms)}}$$

Account for the fact that large files often contain bugs

SimiScore

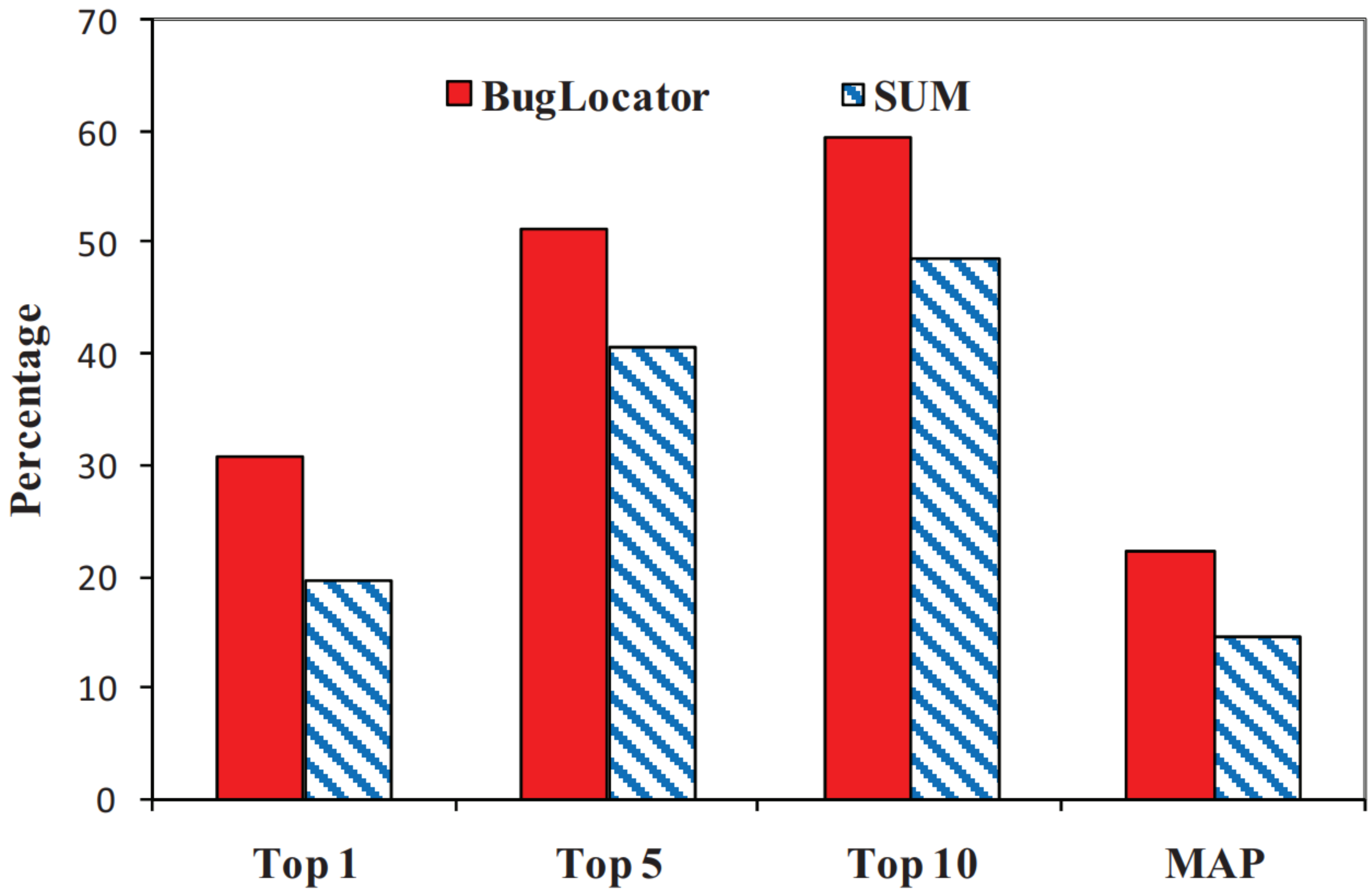


$$SimiScore = \sum_{\substack{\text{All } S_i \text{ that} \\ \text{connect to } F_j}} (Similarity(B, S_i) / n_i)$$

Subject Programs

Project	Description	Study Period	#Fixed Bugs	#Source Files
Eclipse (v3.1)	An open development platform for Java	Oct 2004 - Mar 2011	3075	12863
SWT (v3.1)	An open source widget toolkit for Java	Oct 2004 - Apr 2010	98	484
AspectJ	An aspect-oriented extension to the Java programming language	Jul 2002 - Oct 2006	286	6485
ZXing	A barcode image processing library for Android applications	Mar 2010- Sep 2010	20	391

Results


70
60
50
40
30
20
10
0
Percentage
BugLocator
SUM
Top 1
Top 5
Top 10
MAP

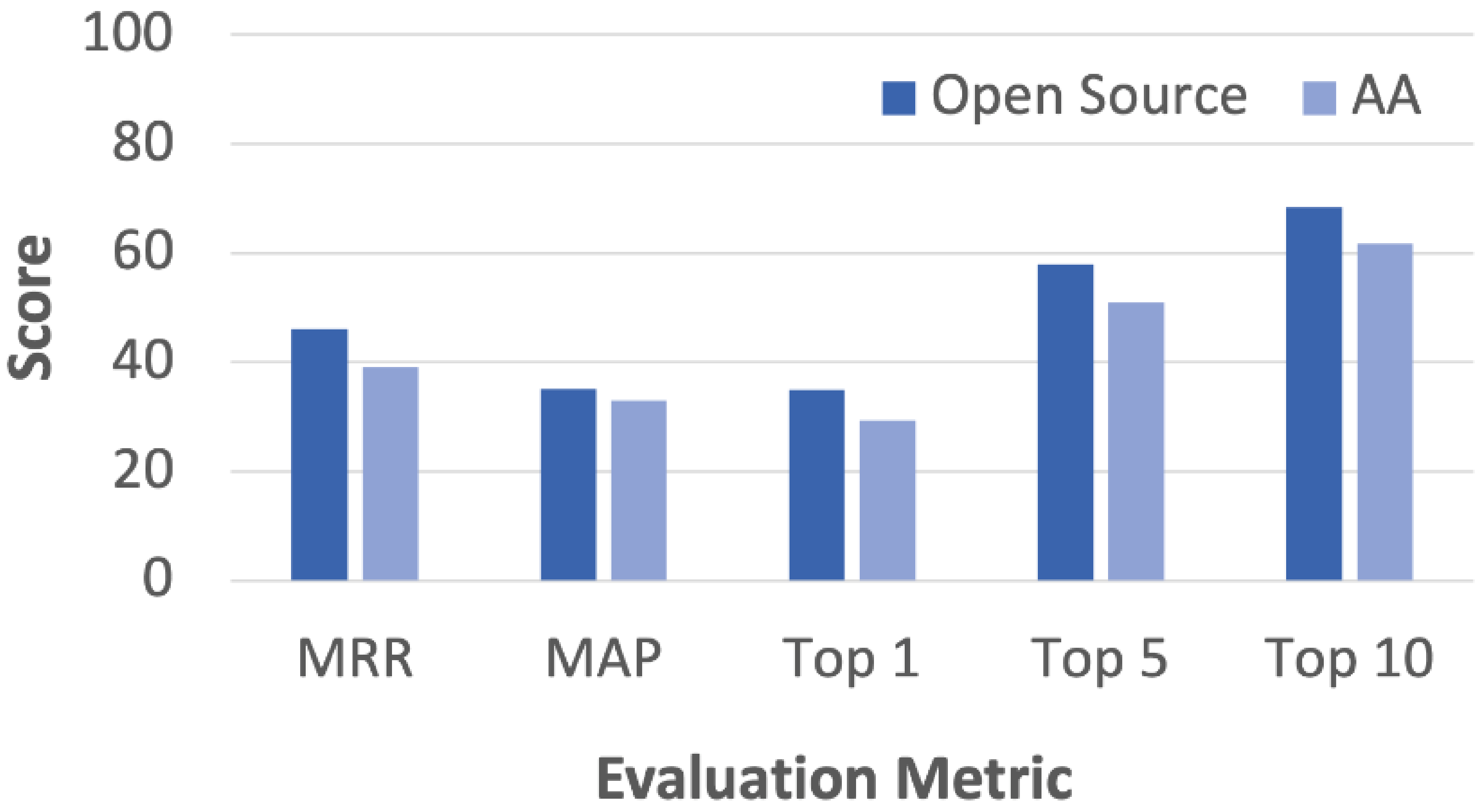
Recent Work with Industry

Legion: Massively Composing Rankers for Improved Bug Localization at Adobe

Darryl Jarman, Jeffrey Berry, Riley Smith, Ferdian Thung, and David Lo

TSE 2022

BugLocator's Performance on Adobe



Adobe's Applicability Requirement

- Consulted 4 developers at Adobe Analytics
- For developers **new** to a repository:
  - the tool should identify a buggy file in the top 10 recommendations at least 70% of the time (*Top 10 score ≥ 70%*)
- For developers **familiar** with a repository
  - the tool should identify a buggy file in the top 5 recommendations at least 80% of the time (*Top 5 score ≥ 80%*)

BL+: Extending BL with Additional Corpora

Pre-processing

SimiScore past comments

FinalScore α values

BugLocator

rVSMScore Corpora Options

- [x] Source code file content
- [ ] Source code differences
- [ ] Commit messages
- [x] Bug report summary
- [ ] Bug report description
- [ ] Bug report past comments

2,772 configurations

Legion: Composing BL Configurations

1. Run all BL configurations

BL+$_{1}$

BL+$_{2111}$

BL+$_{255}$

BL+$_{888}$

BL+$_{599}$

BL+$_{1333}$

BL+$_{1951}$

...

BL+$_{2772}$

2. Generate stacked scores

FinalScore$_{1}$

+

FinalScore$_{2}$

+

FinalScore$_{3}$

+

...

+

FinalScore$_{2772}$

=

Stacked Score

3. Learn a supervised model

FinalScore$_{1}$

FinalScore$_{3}$

FinalScore$_{2}$

...

...

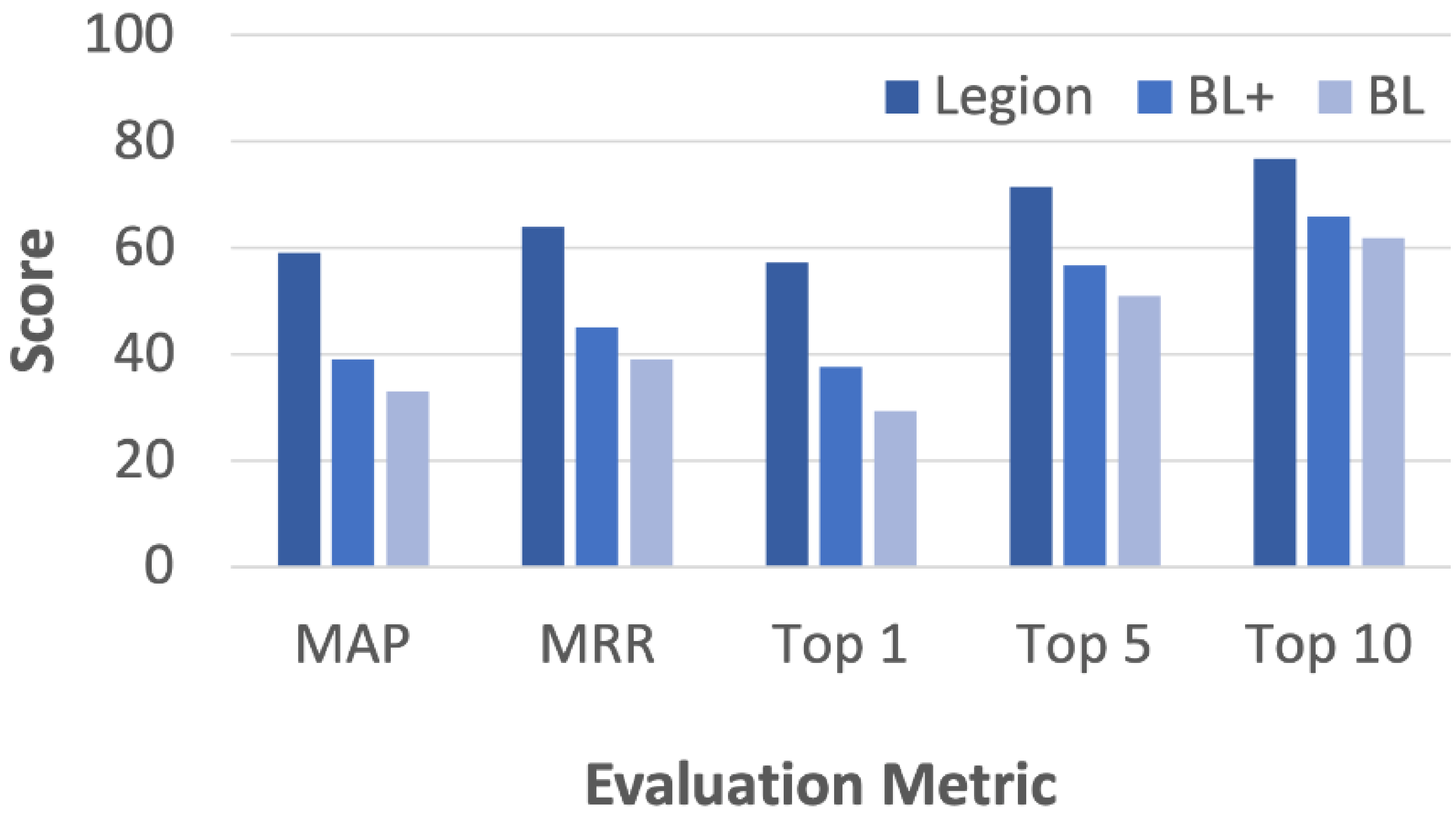
FinalScore$_{2772}$

Stacked Score

↓

Random Forest

Legion Performance



Other Works

Network-Clustered Multi-Modal Bug Localization

Thong Hoang, Richard J. Oentaryo, Tien-Duy B. Le, and David Lo
School of Information Systems, Singapore Management University
{vdthoang.2016, roentaryo, btdle.2012, davidlo}@smu.edu.sg

TSE 2019

Deep Transfer Bug Localization

Xuan Huo, Ferdian Thung, Ming Li, *Member, IEEE*, David Lo, *Member, IEEE*, and Shu-Ting Shi

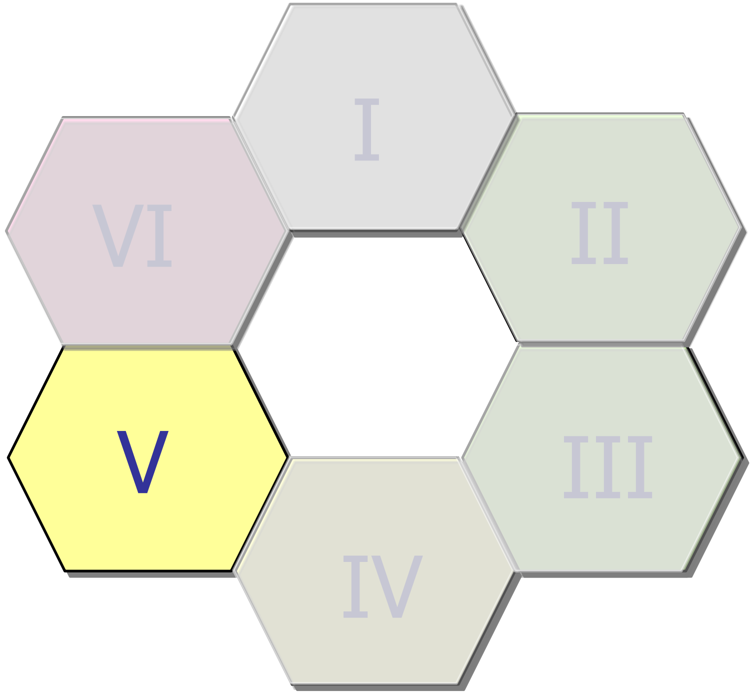
TSE 2021

SMU SINGAPORE MANAGEMENT UNIVERSITY

School of Computing and Information Systems

Outline

Motivation



Data

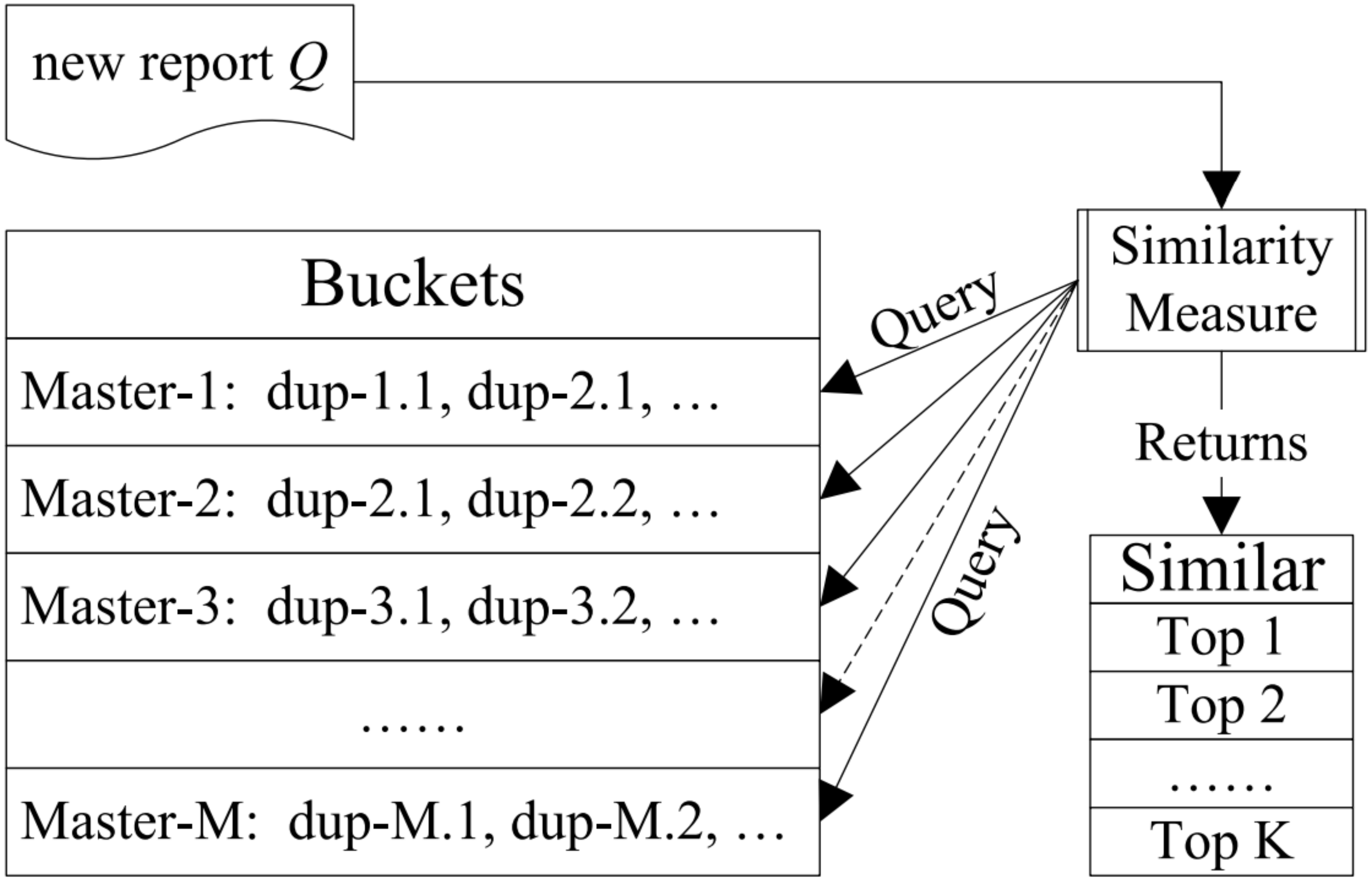
Tasks

Bug Localization

Duplicate Detection

Open Challenges

IV. Duplicate Bug Report Detection



Duplicate Bug Report Detection (DBRD) Process

Popular Early Work

ASE 2011

Towards More Accurate Retrieval of Duplicate Bug Reports

Chengnian Sun*, David Lo†, Siau-Cheng Khoo*, Jing Jiang†
*School of Computing, National University of Singapore
†School of Information Systems, Singapore Management University
suncn@comp.nus.edu.sg, davidlo@smu.edu.sg, khoosc@comp.nus.edu.sg, jingjiang@smu.edu.sg

Most Cited Research Paper of ASE 2011

REP: Lightweight, Learning-Based DBRD

- We want to design an approach that can *learn* from historical data
- The approach needs to consider specific *properties* of bug reports
- The approach needs to be *lightweight* enough

Field	**Description**
Summ	*Summary*: concise description of the issue
Desc	*Description*: detailed outline of the issue, such as what is the issue and how it happens
Prod	*Product*: which product the issue is about
Comp	*Component*: which component the issue is about
Vers	*Version*: the version of the product the issue is about
Prio	*Priority*: the priority of the report, *i.e.*, *P1*, *P2*, *P3*, $\cdots$
Type	*Type*: the type of the report, *i.e.*, *defect*, *task*, *feature*

REP: Lightweight Learning-Based DBRD

$$REP(d,q) = \sum_{i=1}^{7} w_i \times feature_i$$

$$\begin{cases} 1, & \text{if } d.prod = q.prod \\ 0, & \text{otherwise} \end{cases}$$

$$\begin{cases} 1, & \text{if } d.comp = q.comp \\ 0, & \text{otherwise} \end{cases}$$

$$\begin{cases} 1, & \text{if } d.type = q.type \\ 0, & \text{otherwise} \end{cases}$$

$$\frac{1}{1 + |d.prio - q.prio|}$$

$$\frac{1}{1 + |d.vers - q.vers|}$$

19 parameters *tuned* based on **historical** human decisions on training data

$$\sum_{t \in d \cap q} IDF(t) \times \frac{TF_D(d,t)}{k_1 + TF_D(d,t)} \times W_Q$$

$$\text{where } \boxed{W_Q = \frac{(k_3 + 1) \times TF_Q(q,t)}{k_3 + TF_Q(q,t)}}$$

$BM25F_{ext}(d,q)$ //of unigrams

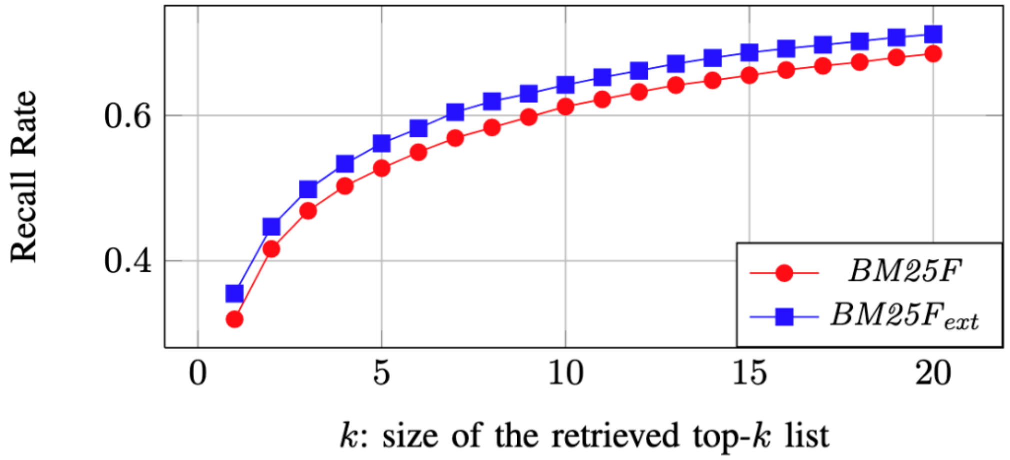
$BM25F_{ext}(d,q)$ //of bigrams

Modification: BM25F is designed for short queries, while bug reports can be long

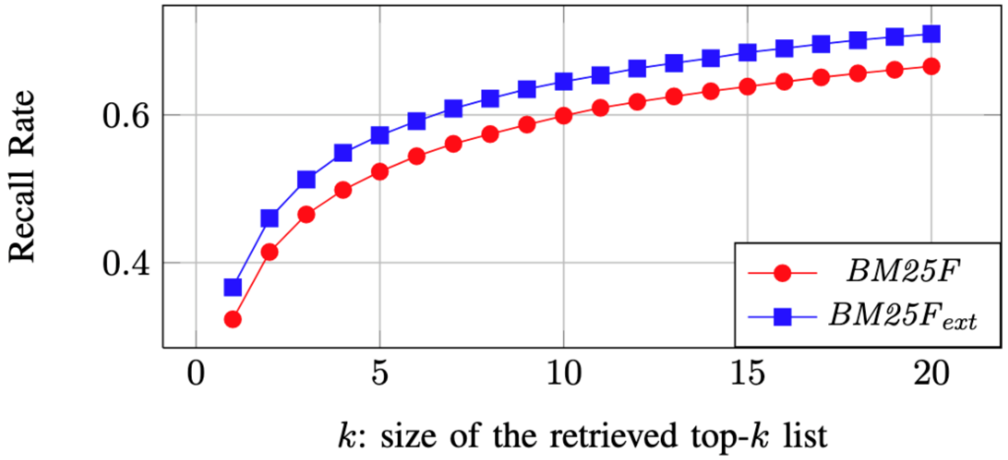
Experiment - Dataset

Dataset	Size	Period		Training Reports		Testing Reports	
		From	To	#Duplicate	#All	#Duplicate	#All
OpenOffice	31,138	2008-01-01	2010-12-21	200	3,696	3,171	27,442
Mozilla	75,653	2010-01-01	2010-12-31	200	4,529	6,725	71,124
Eclipse	45,234	2008-01-01	2008-12-31	200	5,863	2,880	39,371
Large Eclipse	209,058	2001-10-10	2007-12-14	200	3,528	27,295	205,530

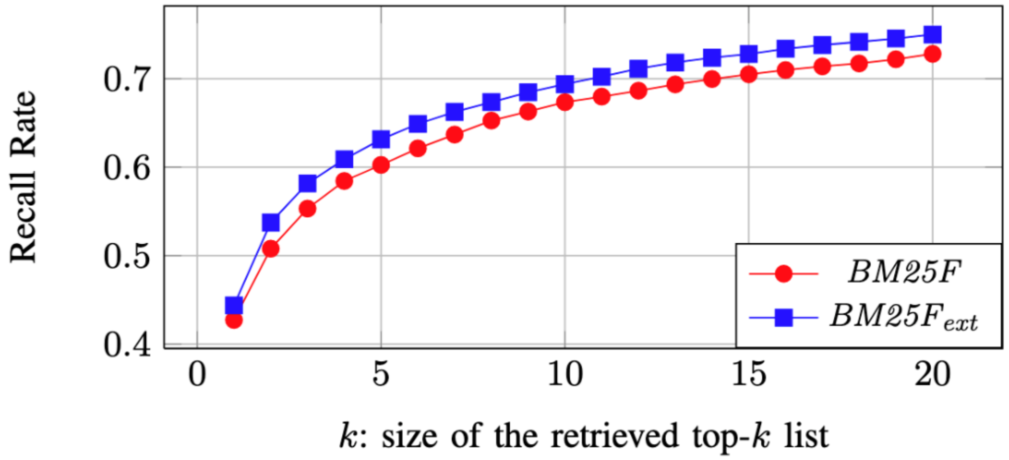
Experiment - Result



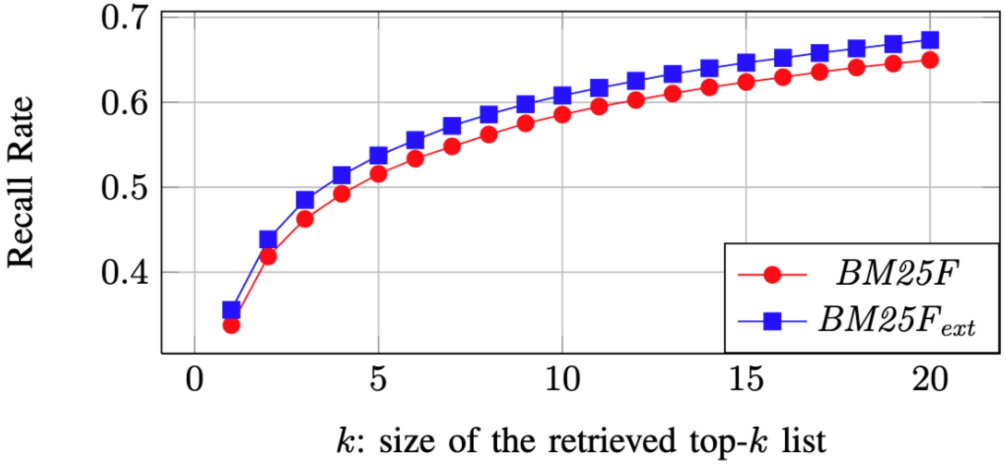
(a) OpenOffice



(b) Mozilla



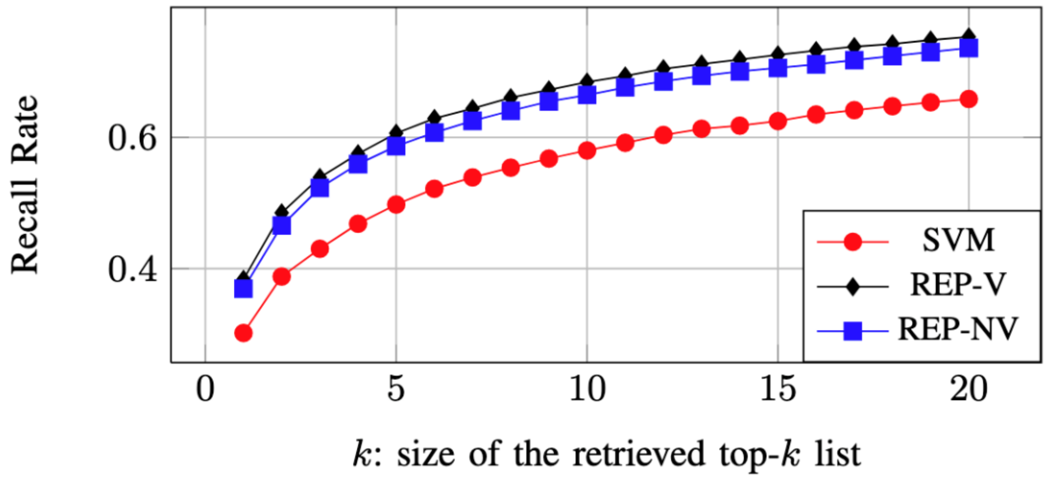
(c) Eclipse



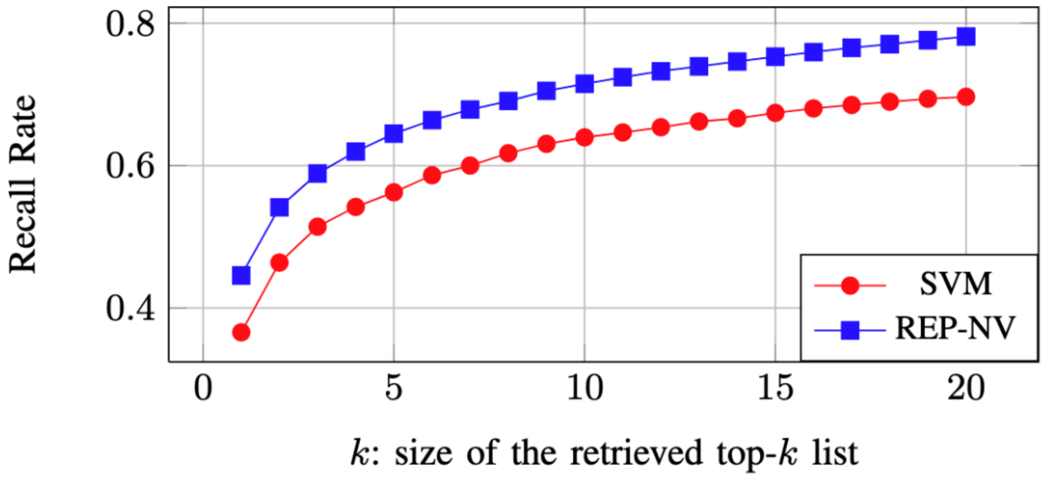
(d) Large Eclipse

$BM25F_{ext}$ is more effective than BM25F

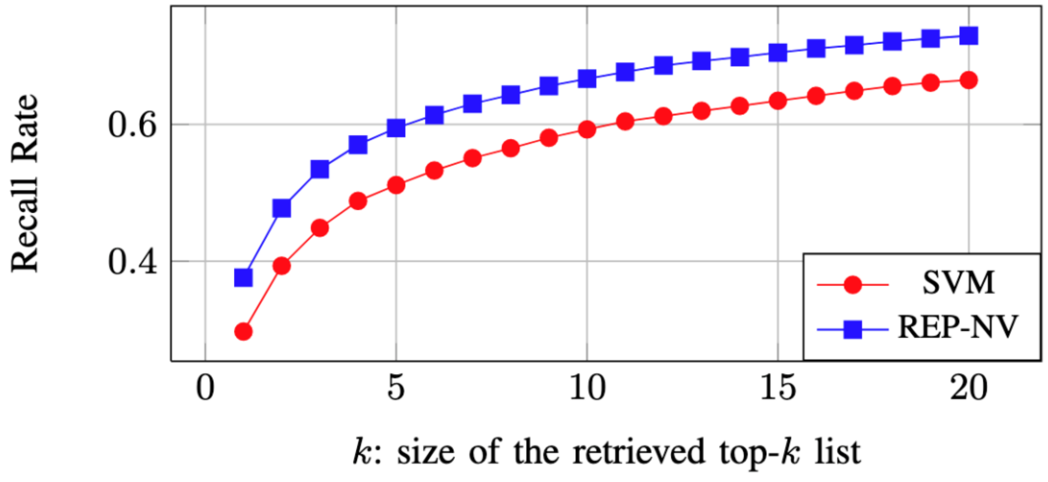
Experiment - Result



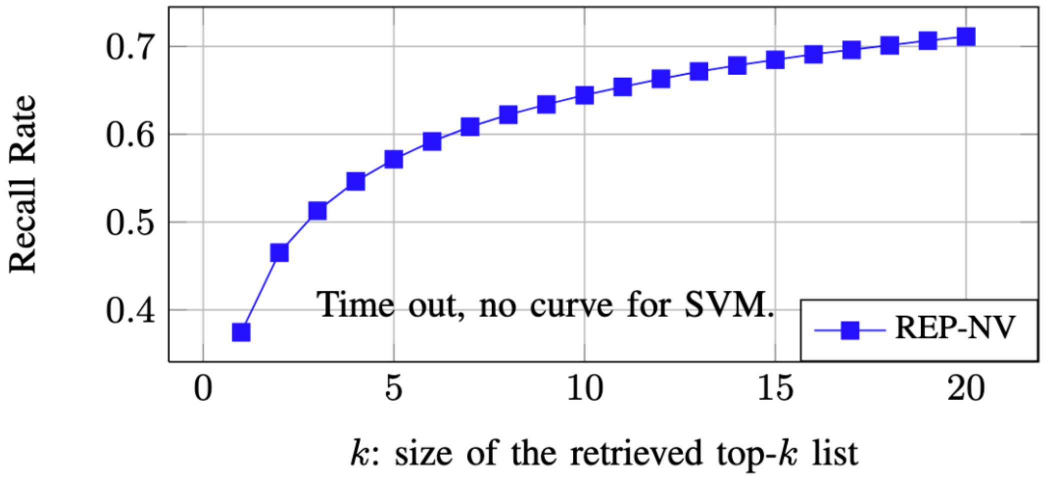
(a) OpenOffice



(c) Eclipse



(b) Mozilla



(d) Large Eclipse

REP is more effective and efficient than ICSE’10 (SVM)

Benchmarking Study

TOSEM 2022

Duplicate Bug Report Detection: How Far Are We?

TING ZHANG, Singapore Management University, Singapore
DONGGYUN HAN, Royal Holloway, University of London, United Kingdom
VENKATESH VINAYAKARAO, Chennai Mathematical Institute, India
IVANA CLAIRINE IRSAN, Singapore Management University, Singapore
BOWEN XU*, Singapore Management University, Singapore
FERDIAN THUNG, Singapore Management University, Singapore
DAVID LO, Singapore Management University, Singapore
LINGXIAO JIANG, Singapore Management University, Singapore

Motivation

- Limitations of existing datasets: *old* bug reports from *Bugzilla* with *latest status*, e.g., SABD[1] was trained and evaluated on the following dataset

Dataset	Period	Training		Validation		Test			Total
		Duplicate	All	Duplicate	All	Start Date	Duplicate	All	
Eclipse	10/10/01 - 31/12/08	27,481	198,183	1,446	14,703	01/01/08	4,380	45,794	258,680
Mozilla	07/04/98 - 31/12/10	122,199	438,806	6,431	44,014	01/01/10	9,701	65,940	548,760
OpenOffice	16/10/00 - 31/12/10	13,570	80,786	714	4,109	01/01/08	4,664	31,333	116,228
Netbeans	21/08/98 - 31/12/08	16,639	116,351	875	5,548	01/01/08	5,009	31,667	153,566

- Lack of comparison among
  - Recent research tools, e.g., SABD[1], DC-CNN[2], HINDBR[3]
  - Research and industrial tools

[1] Rodrigues, Irving Muller, Daniel Aloise, Eraldo Rezende Fernandes, and Michel Dagenais. "A soft alignment model for bug deduplication." In *Proceedings of the 17th International Conference on Mining Software Repositories*, pp. 43-53. 2020.
[2] He, Jianjun, Ling Xu, Meng Yan, Xin Xia, and Yan Lei. "Duplicate bug report detection using dual-channel convolutional neural networks." In *Proceedings of the 28th International Conference on Program Comprehension*, pp. 117-127. 2020.
[3] Xiao, Guanping, Xiaoting Du, Yulei Sui, and Tao Yue. "Hindbr: Heterogeneous information network based duplicate bug report prediction." In *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, pp. 195-206. IEEE, 2020.

Motivation

We aim to:

- Provide a benchmark that addresses the limitations of existing datasets
- Compare research tools on the same dataset
- Compare research and industrial tools

vscodebot bot commented on Aug 27, 2021

(Experimental duplicate detection)
Thanks for submitting this issue. Please also check if it is already covered by an existing one, like:

- issue (#130958)
- bugs (#130911)
- setting is not work (#127954)
- Terminal not working (#129507)
- Cannot find file (#129006)

Research Questions

RQ1: How significant are the potential biases on the evaluation of DBRD techniques?

RQ2: How do state-of-the-art DBRD research tools perform on recent data from diverse ITSs?

RQ3: How do the DBRD approaches proposed in research literature compare to those used in practice?

Research Questions

RQ1: How significant are the potential biases on the evaluation of DBRD techniques?

RQ2: How do state-of-the-art DBRD research tools perform on recent data from diverse ITSs?

RQ3: How do the DBRD approaches proposed in research literature compare to those used in practice?

RQ1 - Dataset

Project	Age	Train		Test	Total	
		# BRs (% Dup)	# Dup Pairs	# BRs (% Dup)	# BRs (% Dup)	# Master BRs
Mozilla	Old	198,653 (9.9%)	35,474	139,502 (9.9%)	338,155 (9.9%)	21,554
	Recent	137,886 (10.1%)	60,498	55,701 (11.2%)	193,587 (10.4%)	10,702
Eclipse	Old	49,355 (5.5%)	4,482	25,021 (12.1%)	74,376 (7.7%)	3,254
	Recent	19,607 (4.7%)	1,725	7,976 (6.5%)	27,583 (5.2%)	959

Age bias: Statistics of old (2012–2014) and recent (2018–2020) data

Platform	Summary	Description	Product	Component	Priority	Severity	Version
Eclipse	10.8%	-	7.8%	11.7%	1.2%	5.6%	8.6%
Mozilla	11.8%	-	21.4%	24.5%	24.5%	5.4%	4.2%

State bias: The percentage of BRs changed the corresponding state in 2018–2020

RQ1 - Dataset

ITS	Project	Train		Test	Total
		# BRs (% Dup)	# Dup Pairs	# BRs (% Dup)	# BRs (% Dup)
Bugzilla	Eclipse	19,607 (4.7%)	1,725	7,976 (6.5%)	27,583 (5.2%)
	Mozilla	137,886 (10.1%)	35,474	55,701 (11.2%)	193,587 (10.4%)
Jira	Hadoop	10,276 (2.8%)	328	3,740 (2.5%)	14,016 (2.7%)
	Spark	6,738 (4%)	414	2,841 (3%)	9,579 (3.7%)
GitHub	Kibana	9,849 (2.9%)	376	7,167 (2.6%)	17,016 (2.8%)
	VSCode	40,801 (7.2%)	9,008	21,291 (6.8%)	62,092 (7%)

ITS bias: Statistics of data in the six projects

RQ1 - Result

Mann-Whitney-U with Cliff's Delta Effect Size $|d|$ Results

Bias	Approach	Data	p-value	$\|d\|$
Age	REP	Eclipse	0.003	0.78 (large)
		Mozilla	0.005	0.72 (large)
	Siamese Pair	Eclipse	< 0.001	1 (large)
		Mozilla	0.003	0.76 (large)
	SABD	Eclipse	0.001	0.82 (large)
		Mozilla	0.012	0.66 (large)
State	REP	Eclipse	0.105	0.44 (medium)
		Mozilla	0.190	0.36 (medium)
	Siamese Pair	Eclipse	0.063	0.5 (large)
		Mozilla	0.190	0.36 (medium)
	SABD	Eclipse	0.315	0.28 (small)
		Mozilla	0.315	0.28 (small)
ITS	REP	Jira	0.056	0.36 (medium)
		GitHub	< 0.001	0.66 (large)
	Siamese Pair	Jira	< 0.001	1 (large)
		GitHub	< 0.001	0.97 (large)
	SABD	Jira	< 0.001	0.97 (large)
		GitHub	< 0.001	0.77 (large)

Answer:

Age Bias and **ITS Bias** have a statistically significant impact, while **State Bias** does not.

Research Questions

RQ1: How significant are the potential biases on the evaluation of DBRD techniques?

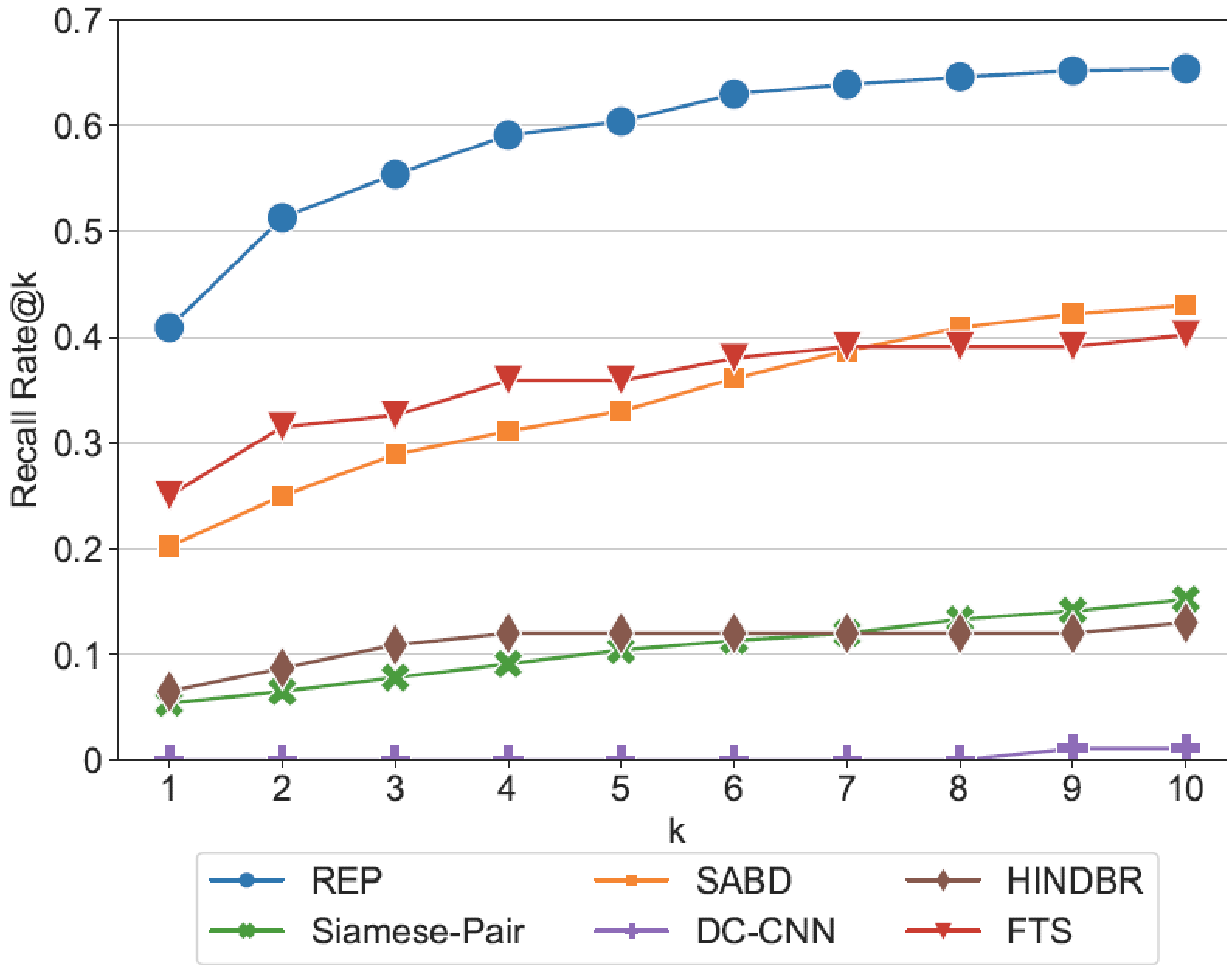
RQ2: How do state-of-the-art DBRD research tools perform on recent data from diverse ITSs?

RQ3: How do the DBRD approaches proposed in research literature compare to those used in practice?

RQ2 – Result

Lightweight & overall best performer in 2022, over various industry and research tools on recent diverse datasets

Hadoop



Answer:

Overall, REP performs the best, especially for typical bug repositories with <10k bug reports.[1]

[1] Average number of issues in 994 repositories: 2,365: Joshi, Saket Dattatray, and Sridhar Chimalakonda. "Rapidrelease-a dataset of projects and issues on github with rapid releases." In *MSR 2019.*

Research Questions

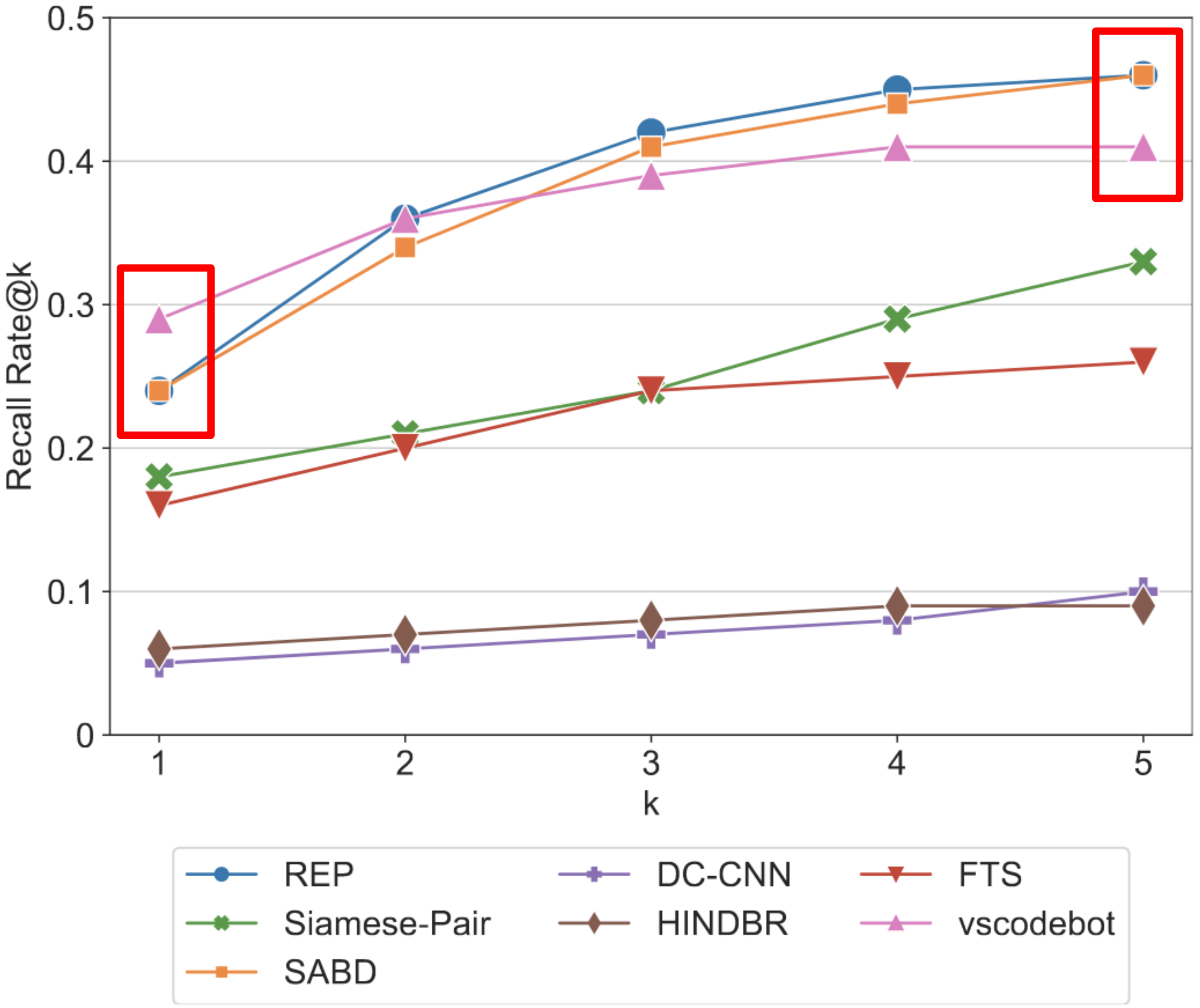
RQ1:How significant are the potential biases on the evaluation of DBRD techniques?

RQ2: How do state-of-the-art DBRD research tools perform on recent data from diverse ITSs?

RQ3: How do the DBRD approaches proposed in research literature compare to those used in practice?

RQ3 - VSCode dataset

Recall Rate@k comparing the tools in research and in practice on the VSCode data



Our Recent Work

Can we do better?

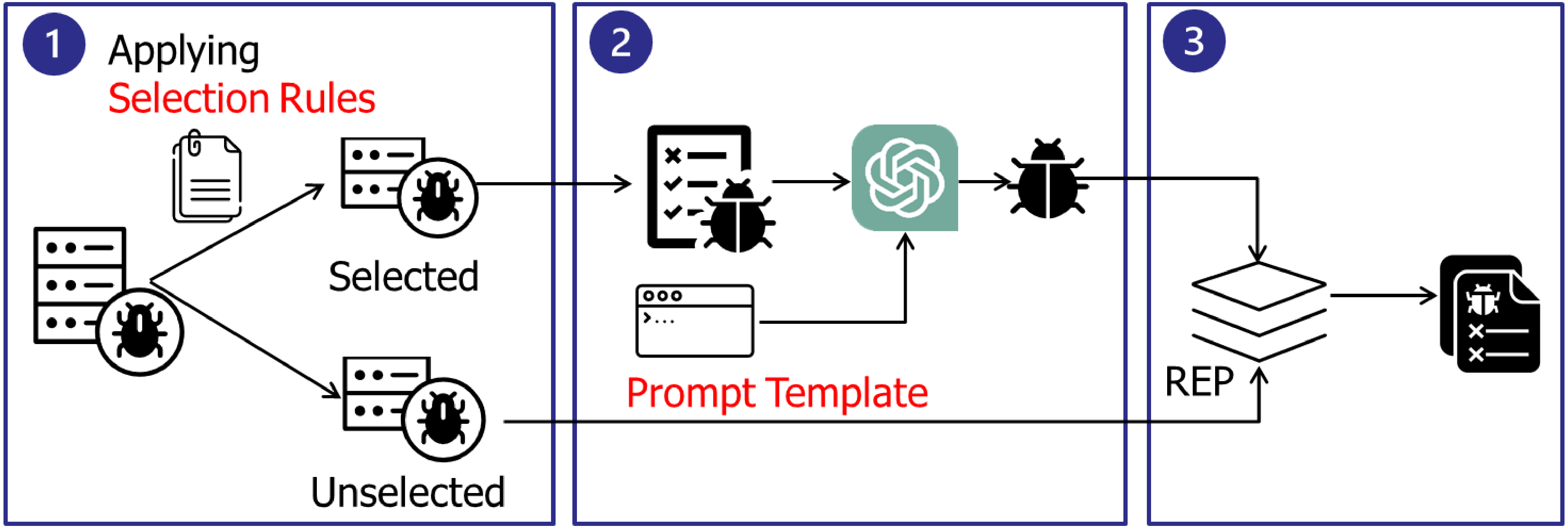
Combining REP and ChatGPT for better DBRD

Motivation

CHAT GPT

ChatGPT is latest advanced generative AI technique

Process


1
Applying
Selection Rules
Selected
Unselected
2
Prompt Template
3
REP

Methodology

- Selection Rules to select bug reports for ChatGPT to process:
  - *Length*: long bug reports
  - *Content*: complex structure
- Prompt template:

Prompt Template:

```
I have a bug report which contains summary and
↪ description. I want you to select keywords
↪ from both parts which keep the main meaning
↪ of the bug report. These keywords would be
↪ used for duplicate bug report detection.
↪ Output format: `Summary: Selected Keywords
↪ \n Description: Selected Keywords` \n\n >>>
↪ Summary: [Summary] \n\n >>> Description:
[Description]
```

Results

Dataset	Total Bugs	Training Pairs	Validation Pairs	Testing	
				# of Duplicate Bugs	# Run By ChatGPT
Spark	9,579	626	26	81	59 (72%)
Hadoop	14,016	626	26	92	57 (62%)
Kibana	17,016	724	28	184	114 (62%)

Results

Dataset	RR@1	RR@3	RR@5	RR@10	Improv. Over SOTA
Spark	0.346	0.432	0.481	0.593	6.7%
Hadoop	0.391	0.565	0.609	0.652	7%
Kibana	0.408	0.571	0.62	0.674	8.7%

Hadoop Common / HADOOP-16795 Java 11 compile support / HADOOP-17091

[JDK11] Fix Javadoc errors

Description

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  17.982 s
[INFO] Finished at: 2020-06-20T01:56:28Z
[INFO] ------------------------------------------------------------------------
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-javadoc-plu
[ERROR] Exit code: 1 - javadoc: warning - You have specified the HTML ver
[ERROR] The default is currently HTML5 and the support for HTML 4.01 will
[ERROR] in a future release. To suppress this warning, please ensure that
[ERROR] in your comments are valid in HTML5, and remove the -html4 option
[ERROR] /home/jenkins/jenkins-slave/workspace/hadoop-multibranch_PR-2084/
[ERROR]       com.google.protobuf.GeneratedMessageV3 implements
[ERROR]                                    ^
[ERROR]   symbol:   class GeneratedMessageV3
[ERROR]   location: package com.google.protobuf
[ERROR] /home/jenkins/jenkins-slave/workspace/hadoop-multibranch_PR-2084/
[ERROR]         com.google.protobuf.GeneratedMessageV3 implements
[ERROR]                                    ^
[ERROR]   symbol:   class GeneratedMessageV3
[ERROR]   location: package com.google.protobuf
[ERROR] /home/jenkins/jenkins-slave/workspace/hadoop-multibranch_PR-2084/
```

ChatGPT keywords: Javadoc, HTML version, HTML4, HTML5, warning, comments, valid, GeneratedMessageV3, package, not found, error

Retrieved Master Bug Report by REP

Description

Builds are failing in PR with following exception in native client.

```
[WARNING] make[2]: Leaving directory '/home/jenkins/jenkins-slave/workspa
[WARNING] /opt/cmake/bin/cmake -E cmake_progress_report /home/jenkins/jen
[WARNING] [ 28%] Built target common_obj
[WARNING] make[2]: Leaving directory '/home/jenkins/jenkins-slave/workspa
[WARNING] /opt/cmake/bin/cmake -E cmake_progress_report /home/jenkins/jen
[WARNING] [ 28%] Built target gmock_main_obj
[WARNING] make[1]: Leaving directory '/home/jenkins/jenkins-slave/workspa
[WARNING] Makefile:127: recipe for target 'all' failed
[WARNING] make[2]: *** No rule to make target '/home/jenkins/jenkins-slav
[WARNING] make[1]: *** [main/native/libhdfspp/lib/proto/CMakeFiles/proto_
[WARNING] make[1]: *** Waiting for unfinished jobs....
[WARNING] make: *** [all] Error 2
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary:
[INFO]
[INFO] Apache Hadoop Main ................................. SUCCESS [  0.
[INFO] Apache Hadoop Build Tools .......................... SUCCESS [  1.
[INFO] Apache Hadoop Project POM .......................... SUCCESS [  0.
[INFO] Apache Hadoop Annotations .......................... SUCCESS [  1.
[INFO] Apache Hadoop Project Dist POM ..................... SUCCESS [  0.
[INFO] Apache Hadoop Assemblies ........................... SUCCESS [  0.
[INFO] Apache Hadoop Maven Plugins ........................ SUCCESS [  4.
```

Creating this ticket, as couple of pull requests had the same issue.

e.g https://builds.apache.org/job/hadoop-multibranch/job/PR-1591/2/artifact/out/patch-compile-root.txt
https://builds.apache.org/job/hadoop-multibranch/job/PR-1614/1/artifact/out/patch-compile-root.txt

Correct Master Bug Report

Description

This issue is to run mvn `javadoc:javadoc` successfully in Apache Hadoop with Java 11.
Now there are many errors.

Summary of **Yellow** Segment (Parts IV, V)

- Bug localization
  - Leverage multiple notions of *similarity*
  - *Diverse artifacts* in repos can be used
  - Meaning of similarity can be *learned* from history
- Duplicate bug report detection
  - *Historical* data can be used to tune detectors
  - *Biases* can affect experiment results
  - *LLM* can be helpful

IV

Localization

V

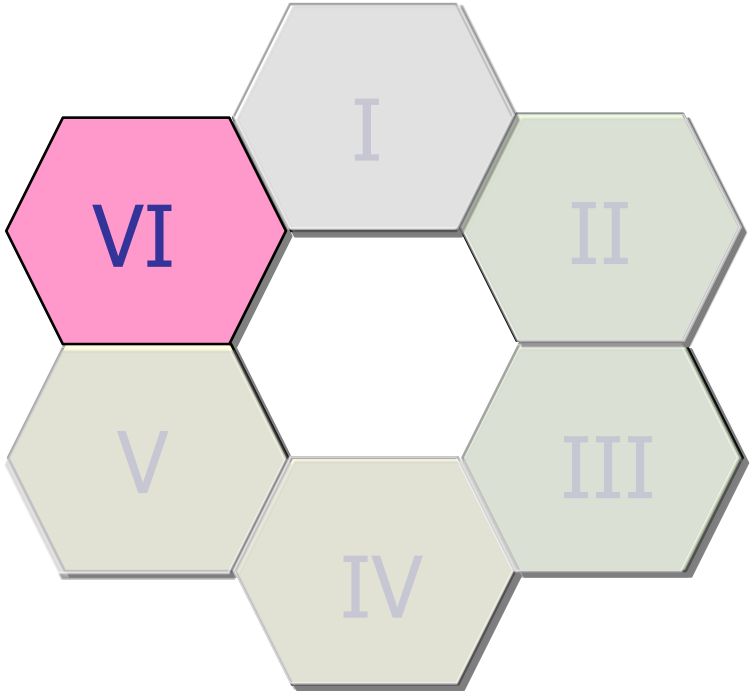
Duplicate

SMU SINGAPORE MANAGEMENT UNIVERSITY

School of Computing and Information Systems

Outline

Motivation



Data

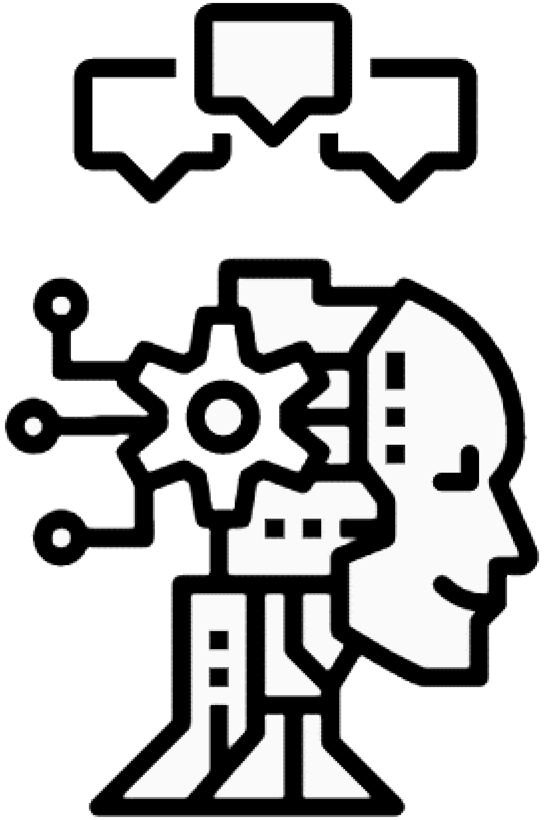
Tasks

Bug Localization

Duplicate Detection

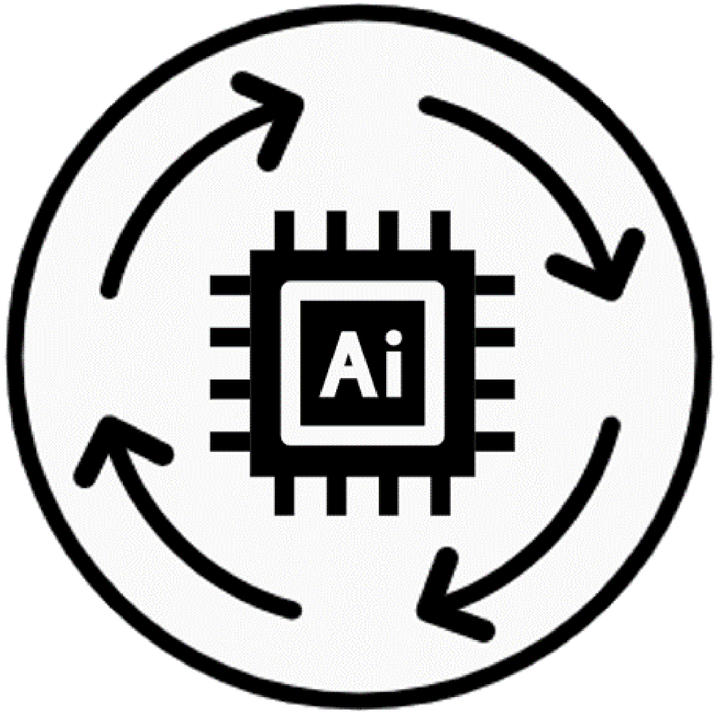
Open Challenges

Open Challenges I



Explainable Automated Bug Report Management

Open Challenges II



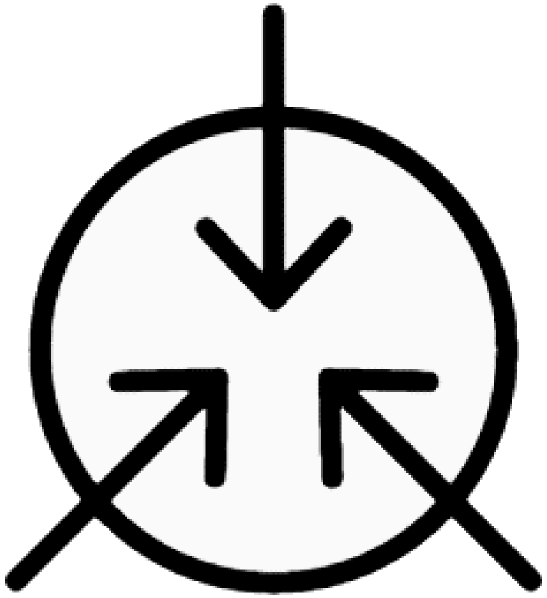
Tight **Integration** with Developer Workflow

vscodebot bot commented on Aug 27, 2021 ...

(Experimental duplicate detection)
Thanks for submitting this issue. Please also check if it is already covered by an existing one, like:

- issue (#130958)
- bugs (#130911)
- setting is not work (#127954)
- Terminal not working (#129507)
- Cannot find file (#129006)

Open Challenges III



ID	Category
T1	Bug localization
T2	Bug assignment
T3	Duplicate/similar bug detection
T4	Bug categorization
T5	Bug fixing time prediction
T6	Bug severity/priority prediction
T7	Bug report completion/refinement
T8	Bug-commit linking
T9	Bug report summarization/visualization
T10	Re-opened bug prediction

Holistic Analysis
of Multiple Tasks

Open Challenges III

Information Retrieval Based Nearest Neighbor Classification for Fine-Grained Bug Severity Prediction

Yuan Tian[1], David Lo[1], and Chengnian Sun[2]
[1]Singapore Management University, Singapore
[2]National University of Singapore, Singapore
{yuan.tian.2011,davidlo}@smu.edu.sg, suncn@comp.nus.edu.sg

WCRE 2012

"Duplicate bug reports are utilized to determine what bug report features, be it textual, ordinal, or categorical, are important."

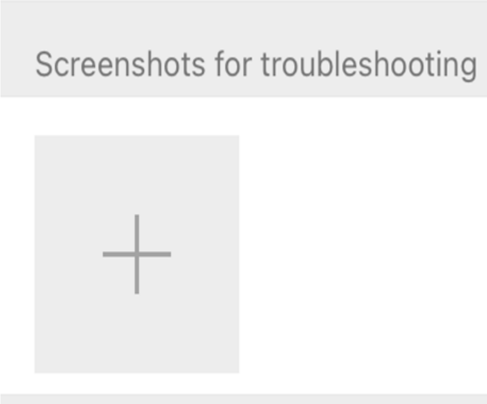
Won Most Influential Paper Award @ SANER 2022

Open Challenges IV

Prioritizing User Feedback from Twitter: A Survey Report

Emitza Guzman   Mohamed Ibrahim   Martin Glinz

CSI-SE 2017



Screen capture tool with video recording

- Add images or videos with voice over to your bug reports
- Add comments directly to the snapshot to provide effective feedback
- Create instant bug report & transfer it to ReQtest
- Get the bug report to the right person – right away!

https://reqtest.com/

Handle **Rich Media**

Open Challenges V

Emerging App Issue Identification from User Feedback: Experience on WeChat

Cuiyun Gao[†], Wujie Zheng[§*], Yuetang Deng[§], David Lo[‡], Jichuan Zeng[†], Michael R. Lyu[†], Irwin King[†]

ICSE 2019

A Machine Learning Approach for Vulnerability Curation

Yang Chen
Veracode
ychen@veracode.com

Andrew E. Santosa
Veracode
asantosa@veracode.com

Ang Ming Yi
Veracode
mang@veracode.com

Abhishek Sharma
Veracode
absharma@veracode.com

Asankhaya Sharma
Veracode
asharma@veracode.com

David Lo
Singapore Management University
davidlo@smu.edu.sg

MSR 2020

Won ACM SIGSOFT Distinguished Paper Award

Industrial Collaborations

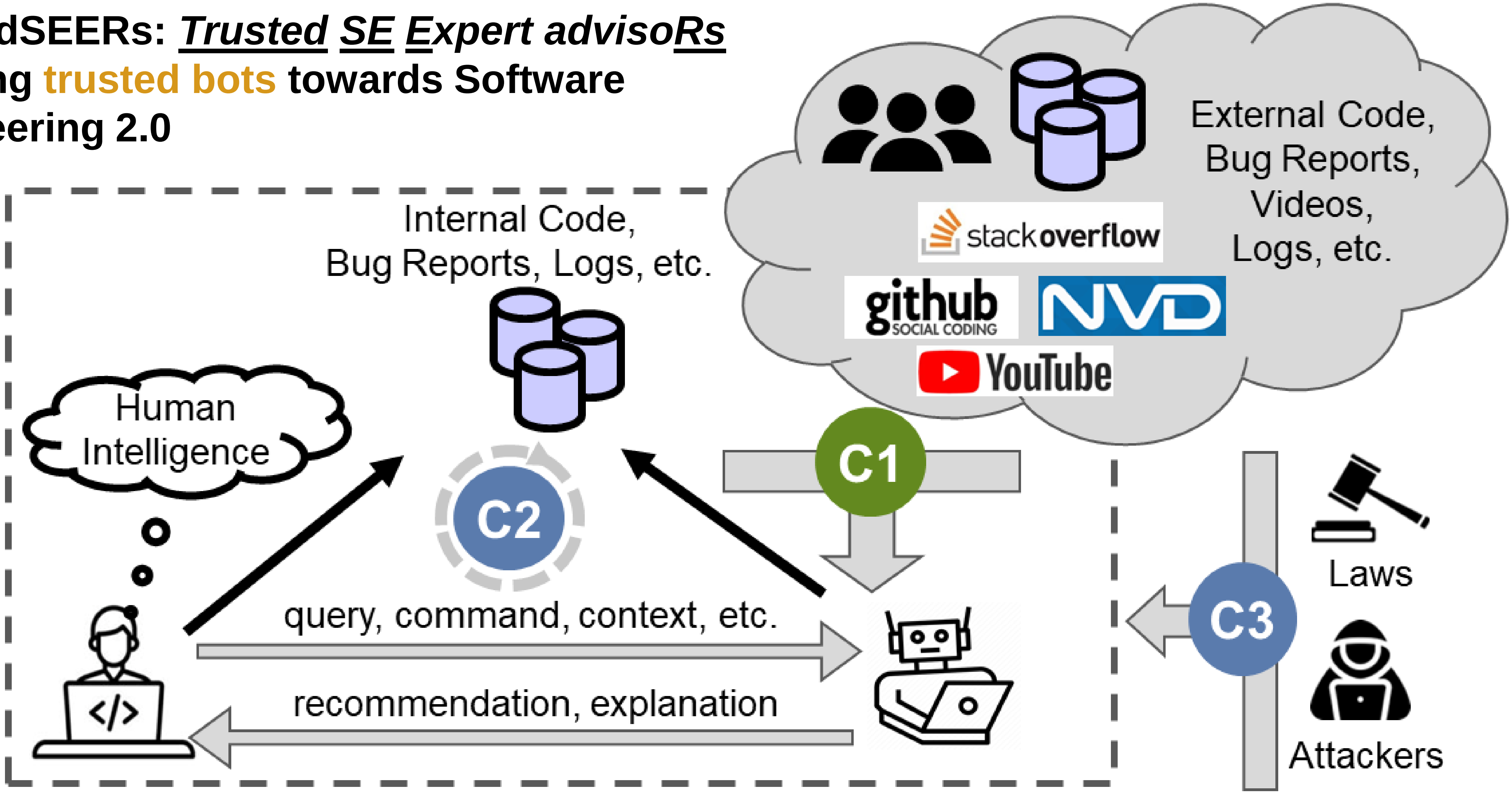
Open Challenges VI

Automating App Review Response Generation

Cuiyun Gao[†] Jichuan Zeng[†] Xin Xia[‡] David Lo[§] Michael R. Lyu[†] Irwin King[†]
[†]Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, China
[‡]Faculty of Information Technology, Monash University, Australia
[§]School of Information Systems, Singapore Management University, Singapore
{cygao,jczeng,lyu,king}@cse.cuhk.edu.hk xin.xia@monash.edu davidlo@smu.edu.sg

ASE 2019

TrustedSEERs: *Trusted SE Expert advisoRs*
Building trusted bots towards Software Engineering 2.0



SMU SINGAPORE MANAGEMENT UNIVERSITY

NRF Investigatorship project, 2023-2028 ($3.2M)
Individual research grant, similar to ERC Advanced

School of Computing and Information Systems

Thank you!

Questions? Comments? Advice?

davidlo@smu.edu.sg