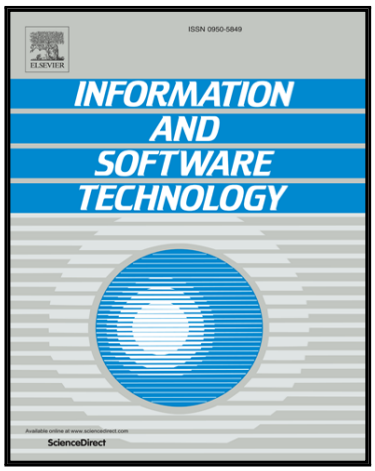


# Just Enough Semantics An Information Theoretic Approach for IR-based Software Bug Localization

Saket Khatiwada, Miroslav Tushev, Anas Mahmoud

PII: S0950-5849(16)30226-9
DOI: 10.1016/j.infsof.2017.08.012
Reference: INFSOF 5870

To appear in: *Information and Software Technology*

Received date: 7 October 2016
Revised date: 22 August 2017
Accepted date: 24 August 2017

Please cite this article as: Saket Khatiwada, Miroslav Tushev, Anas Mahmoud, Just Enough Semantics An Information Theoretic Approach for IR-based Software Bug Localization, *Information and Software Technology* (2017), doi: 10.1016/j.infsof.2017.08.012

# *Just Enough Semantics*
# An Information Theoretic Approach for IR-based Software Bug Localization

Saket Khatiwada, Miroslav Tushev, Anas Mahmoud

*Division of Computer Science and Engineering*
*Louisiana State University*
*Baton Rouge, LA, 70803*
*skhati1@lsu.edu, mtushe1@lsu.edu, mahmoud@csc.lsu.edu*

**Abstract**

**Context:** Software systems are often shipped with defects. Whenever a bug is reported, developers use the information available in the associated report to locate source code fragments that need to be modified in order to fix the bug. However, as software systems evolve in size and complexity, bug localization can become a tedious and time-consuming process. To minimize the manual effort, contemporary bug localization tools utilize Information Retrieval (IR) methods for automated support. IR methods exploit the textual content of bug reports to automatically capture and rank relevant buggy source files.
**Objective:** In this paper, we propose a new paradigm of information-theoretic IR methods to support bug localization tasks in software systems. These methods, including Pointwise Mutual Information (PMI) and Normalized Google Distance (NGD), exploit the co-occurrence patterns of code terms in the software system to reveal hidden textual semantic dimensions that other methods often fail to capture. Our objective is establish accurate semantic similarity relations between source code and bug reports.
**Method:** Five benchmark datasets from different application domains are used to conduct our analysis. The proposed methods are compared against classical IR methods that are commonly used in bug localization research.
**Results:** The results show that information-theoretic IR methods significantly outperform classical IR methods, providing a semantically aware, yet, computationally efficient solution for bug localization in large and complex software systems. (A replication package is available at: http://seel.cse.lsu.edu/data/ist17.zip).
**Conclusions:** Information-theoretic co-occurrence methods provide *"just enough semantics"* necessary to establish relations between bug reports and code artifacts, achieving a balance between simple lexical methods and computationally-expensive semantic IR methods that require substantial amounts of data to function properly.

*Keywords:* Information Retrieval, Bug localization, Information theory

## 1. Introduction

Bug localization is a software engineering activity that is concerned with finding buggy source code segments relevant to a specific bug report [1, 2, 3]. Software systems are often shipped with defects. For a relatively large system, the number of defects may range from hundreds to thousands. For example, in 2009, 4,414 bugs were reported for the Eclipse project [4]. Once a bug is reported, developers resort to the description of the bug, available in the bug report, to locate source code segments that should be modified in order to fix the bug.

Bug localization tasks involve examining an error trace in the software system in order to understand the cause of the error and isolate its relevant buggy code fragments [5]. While such a process can be feasible in smaller systems, analyzing and comprehending relatively large and complex systems can be a tedious and error-prone task. In fact, it has been observed that up to 70% of developer time during maintenance is devoted to program comprehension [6, 7]. In order to minimize this effort, researchers employed a large number of automated methods for software bug localization. The main objective is to partially alleviate the manual effort by helping developers to automatically deal with the growing complexity of source code without compromising the quality of their work.

Automated methods for localizing bugs in software systems can be classified into two main categories, including dynamic and static. The dynamic approach locates bugs by collecting and analyzing program data, breakpoints, or the execution traces of the system [8]. This approach relies on finding differences between the control flows of passing and failing runs of the system under certain input conditions [9, 10, 11]. Evidently, dynamic methods require executing the software, which cannot always be feasible, especially in cases of major errors that extend over several code modules. The static approach, on the other hand, employs Information Retrieval (IR) methods to automatically locate faulty code artifacts. The main assumption is that developers define their identifiers and write their comments in such a way that captures their understanding of the system at the most primitive level [12, 13, 14]. A word in this vocabulary can refer to a domain concept, a system feature, or a design decision, or describe an event, an exception, or an attribute of the system [15, 16, 17]. Such information can be exploited by con-