BugSleuth

Ashish Ramrakhiani

Lakshita Malhotra

INTRODUCTION - Problem

No Fault localization technique is best, no one-size-fits-all solution [1]

Supervised machine learning enhances FL performance but demands an extensive labeled dataset.

[1] D. Zou, J. Liang, Y. Xiong, M. D. Ernst, and L. Zhang, “An empirical study of fault localization families and their combinations,” IEEE Transactions on Software Engineering, vol. 47, no. 2, pp. 332–347, 2019.

INTRODUCTION - Solution

BugSleuth - An unsupervised fault localization technique that combines multiple ranked lists of suspicious statements

Provides an optimal ranked list of suspicious statements thereby localizing bugs in their code

Research Questions

RQ1: How effective is BugSleuth (unsupervised rank aggregation) for localizing defects ?

RQ2: How efficient is using BugSleuth (unsupervised rank aggregation) for fault localization ?

RQ3: How does BugSleuth (unsupervised rank aggregation) compare against state-of-the-art fault localization techniques ?

Key Idea

Develop an unsupervised fault localization technique that combines the results of various fault localization techniques.

Construct an ordered rank list which minimizes the sum of distances between the input lists and the aggregated list.

Key Idea - Design

n input rank lists of size k ordered by suspicion scores

Rank Aggregator

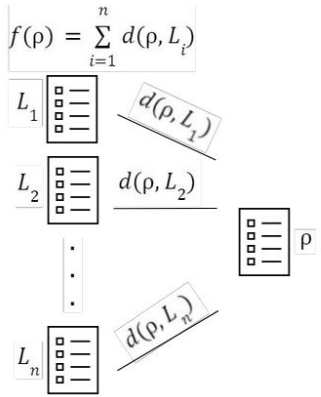
Optimal rank list of size k (k>=1) statements localized

Ranked lists of suspicious statements for multiple fault localization techniques

- f() - objective function
- ρ – combined list
- Li – input lists
- d() - distance function

Optimization:

Choose f(ρ) which has smallest value to obtain an optimal list which is closest to all input lists



Evaluation Plan

Dataset

Defects4J (v2.0) benchmark to evaluate our FL technique.

Defects4J consists of 835 reproducible defects from 17 large open-source Java projects.

Each defect comes with :

(1) one buggy and one developer-repaired version of the project code

(2) a set of developer written tests, all of which pass on the developer-repaired version and at least one of which evidences the defect by failing on the buggy version and

(3) defect information

Ground Truth created by documenting the developer added/modified/deleted lines for 815 defects in Defects4J. We will fine tune our technique on 111 JacksonDatabind defects of Defects4J (v2.0)

Evaluation Plan

Metrics

We will use two metrics, common to FL evaluations

Top-N is the number of defects localized in the top-n ranked statements

- It tells us how efficient our FL technique is in localizing bugs
- Top-1, Top-3 and Top-5 utilized

EXAM is the fraction of ranked statements one must inspect before finding a buggy statement.

- EXAM tells us how high the buggy statements are ranked
- saves practitioner's time and resources otherwise spent on manual bug detection

Thank You