BoostNSift: Implementation of BoostNSift Method Level Bug Localization Technique

# Introduction

BoostNSift is a new, method-level, Information-Retrieval based bug localization technique. It exploits the important information in queries by ‘boost’ing that information as bug localization starts and then ‘sift’s the identified code elements, based on a novel technique that emphasizes the code elements’ specific relatedness to a bug report over its generic relatedness to all bug reports.

# 1 Reproducing BoostNSift with Provided Dataset

To reproduce BoostNSift on provided dataset, perform the following steps:

## Pre-requisites

### (A) Software Installation

The following software must be installed on your system:

1. Java 8 JDK (or above) : <https://www.oracle.com/java/technologies/javase-downloads.html>
2. A Java supported IDE (In the steps below we use Eclipse IDE for Java Developer 2020-09)
3. A latest version of PyCharm: <https://www.jetbrains.com/pycharm/download/#section=windows>. For more information on PyCharm installation follow the following link: <https://www.jetbrains.com/help/pycharm/installation-guide.html>

### (B) Getting your System Ready

1. Make sure that your system has the environment variable PATH for java and JAVA\_HOME are set. For more detail on setting environment variables for Java, follow the following link: <https://java.com/en/download/help/path.html>
2. Download BoostNSift from following link: <https://github.com/razi-rechan/boostNsift/blob/master/BoostNSift.zip> and unzip it to eclipse-workspace folder. If you are using Windows operating system, eclipse-workspace should be located at C:\Users\ (Your Computer Name)\eclipse-workspace
3. Download the sample dataset provided: <https://github.com/razi-rechan/boostNsift/blob/master/Dataset.zip> and unzip it on your preferred location at your system. Note paths under unzipped Dataset folder will be used in next steps.

## Executing BoostNSift

To execute BoostNSift, perform the following steps:

### (A) Importing BoostNSift into Eclipse IDE

1. Open the Eclipse IDE
2. From the “File” Menu select “Import”
3. From the Import Dialog select “General 🡪 Existing Projects into Workspace” and click “Next”
4. Under the “Import Projects” Dialog, choose “Select root directory” radio button and click “Browse…” button
5. Browse to BoostNSift folder unzipped in eclipse-workspace and click “Select Folder”
6. Under the “Import Projects” Dialog, click “Finish”

### (B) Building Paths for BoostNSift Project

### In the “Package Explorer” window in Eclipse, right click on BoostNSift folder and from the menu that appears select “Build Path 🡪 Configure Build Path”

1. In “Properties for BoostNSift” Dialog ensure that “Java Build Path" in the left-hand side menu is selected and then click “Libraries” tab in the right-hand side area that appears
2. Under Libraries, expand “Classpath” and select all .jar files, then click on “Remove”, which is located at right-hand side of the Dialog
3. Then, click on “Classpath” once again and click the “Add External JARs…” button
4. Browse to BoostNSift “lib” folder, which is located at ../eclipse-workspace/BoostNSift/lib on your local system, then select all .jar files (press Ctrl+A) and click “Open” button
5. Then, in “Properties for BoostNSift” Dialog, click “Apply and Close” button

### (C) Configuring the Paths in Source File

1. From the “Package Explorer” window in your Eclipse, Open the file “BoostNSift🡪src🡪 boostNsift🡪BoostNSift.java”
2. On line # 47 provide the path to the folder where BoostNsift will build the index of corpus. This could be path to any folder on your system that is dedicated for the use of BoostNSift.
3. On line # 58 provide the pre-processed corpus of a software system in Dataset folder. For example, for SWT software system, corpus can be found on the following path: ..\Dataset\SWT\Corpus\_SWT.csv.
4. Next you need to provide the paths of pre-processed bug titles, descriptions, and comments files at line # 93, 103, and 113, respectively. For example, for SWT software system, these files can be found on the following paths: ..\Dataset\SWT\SWTTitles.csv, ..\Dataset\SWT\SWTDesc.csv, ..\Dataset\SWT\SWTComm.csv, respectively.
5. At line # 124 provide the path of the output file where results will be stored. For example, for SWT software system, you may provide a path: ..\Dataset\SWT\Results\_SWT.csv.

### (D) Running BoostNSift

1. From “Run” Menu click “Run” to execute BoostNSift.

This will generate a file (e.g., Results\_SWT.csv). In the form of a matrix where each row will correspond to a query (i.e., Bug report) and each column will correspond to a source code method, and each cell in matrix will shows the score.

## Assessing BoostNSift

To assess BoostNSift, perform the following steps:

1. Download the two python scripts BoostNSiftEvaluationScript.py and rank\_metrics.py provided from <https://github.com/razi-rechan/boostNsift/blob/master/BoostNSiftEvaluationScript.py> and <https://github.com/razi-rechan/boostNsift/blob/master/rank_metrics.py>, respectively.
2. Place both scripts in the same folder, then right click on each script and open them with PyCharm
3. On line 6, set the methods variable to the number of methods in source code of software system. This is equal to the number of the number lines in corpus file, e.g., ..\Dataset\SWT\Corpus\_SWT.csv.
4. Then on line # 10 in BoostNSiftEvaluationScript.py provide the paths of following four files, respectively:
5. Output file generated in *Executing BoostNSift* step (e.g., Results\_SWT.csv).
6. Answer\_Matrix file, this file can be found in Dataset folder. For example, for SWT software system, answer\_matrix can be found on the following path: ..\Dataset\SWT\AnswerMatrix\_SWT.csv.
7. A path to the file to store complete results. This could be any file in .csv format on your system.
8. A path to the file to store the 10-Files results. This could be any file in .csv format on your system.
9. On line 82, provide the path to the file to store the overall results. For example, for SWT system, you may change that line as follows: np.savetxt("SWT\_IR\_Results.csv", results, fmt='%s')
10. Right click on BoostNSiftEvaluationScript.py file and then click “Run”. Your evaluation results should ready to review in output files.

# 2 Generating New Dataset for BoostNSift

To generate new dataset for testing BoostNSit (i.e., on software systems not provided in our dataset), perform the following steps:

## Generating Corpus

1. Download the source code of a java-based software system. For example, ArgoUML software system can be downloaded from following link: <http://argouml.tigris.org>. Place the source code folder at eclipse workspace. The default location is following: C:\Users\ (Your Computer Name)\eclipse-workspace
2. Download the “CorpusGenerator” project from following link: <https://github.com/razi-rechan/boostNsift/tree/master/Corpus-Generator> and place the project in eclipse workspace.
3. Import the “CorpusGenerator” project into Eclipse IDE following the same steps as defined for importing BoostNSift project, see Section headed: (A) Importing BoostNSift into Eclipse IDE
4. Build path of “CorpusGenerator” project in Eclipse IDE following the same steps as defined for building path for BoostNSift project, see Section headed: (B) Building Paths for BoostNSift Project
5. From package explorer of Eclipse IDE, open the PathsOfSourceFiles.java file (provided in src/corpusGenerator package)
6. Provide the path to the source code folder of the system (e.g., folder of ArgoUML source code downloaded) on line # 14 also provide a name (say SourcePaths.txt) of the output file at line #13.
7. Then, right click PathsOfSourceFiles.java file and Run As java project. It will generate information of all the java files in project.
8. From package explorer of Eclipse IDE, open MainCorpusGenerator.java file (provided in src/corpusGenerator package)
9. Provide the path of file created in 7th step at line # 46 and an output folder (any folder of your choice at your system) name at line # 47.
10. Note you can create parser on method, class, or file level granularity setting by setting args[0] at line # 45. Because, we are interested in method level granularity, hence set it to -methodLevelGranularity.
11. After setting these three parameters run MainCorpusGenerator.java file, by right clicking the file and Run As java application. It will generate corpus in output folder provided.

## Generating Queries (Bug Reports)

Perform the following steps to generate queries (bug reports) for any software system:

1. Open the issue tracking repository of the software system you are generating the new dataset for. For example, for AspectJ software system you can open the following link: <https://www.eclipse.org/aspectj/bugs.php>
2. From repository, download the Titles of bug reports and save them as .csv file at your system
3. From repository, download the Descriptions of bug reports and save them as .csv file at your system
4. From repository, download the Comments of bug reports and save them as .csv file at your system

## Pre-processing the Queries and Corpus

1. Download the “Corpus-PreProcessor” project from following link: <https://github.com/razi-rechan/boostNsift/tree/master/Corpus-PreProcessor> and place the project in eclipse workspace.
2. Import the “Corpus-PreProcessor” project into Eclipse IDE following the same steps as defined for importing BoostNSift project, see Section headed: (A) Importing BoostNSift into Eclipse IDE
3. Import the “Corpus-PreProcessor” project into Eclipse IDE following the same steps as defined for importing BoostNSift project, see Section headed: (A) Importing BoostNSift into Eclipse IDE
4. Build path of “Corpus-PreProcessor” project in Eclipse IDE following the same steps as defined for building path for BoostNSift project, see Section headed: (B) Building Paths for BoostNSift Project
5. From package explorer of Eclipse IDE, open MainCorpusPreProcessor.java file (provided in src/corpusPreprocessor/ package) and provide the path of .corpusRawMethodLevelGranularity file created in generating corpus step (you can find that file in output folder provided in Generating Corpus step). This path will be provided on line # 15.
6. Provide the output folder path for this step at line # 16.
7. Run MainCorpusPreProcessor.java file, by right clicking the file and Run As java application. It will generate the pre-processed corpus in output folder provided in step 6.
8. Repeat step 5-7 for pre-processing bugs titles, descriptions and comments by providing the paths of the respective files.

## Generating Answer Matrix

Answer matrix file contain the information of gold-sets: That is, the ids of the methods resolved for each bug where each row represent each bug, corresponding to the output file, and each cell contains the id of the method resolved against that bug (e.g., see file ..\Dataset\SWT\AnswerMatrix\_SWT.csv). Here bug id is corresponding to the line number where the corpus of the method located in. corpusRawMethodLevelGranularity file. Researchers may create Answer Matrix manually or use some existing repository mining tools (e.g., PyDriller) to get the information of the resolved bugs and then associate them to the bugs for they resolved using information like commit messages. For more detail see our paper Empirical Design section. Please note that Answer-Matrix files for the software systems employed in our study are already provided in archive: <https://github.com/razi-rechan/boostNsift/blob/master/Dataset.zip>.