Towards Detecting Inconsistent Comments in Java Source Code Automatically*

*Nataliia Stulova, Arianna Blasi, Alessandra Gorla, Oscar Nierstrasz

Software Composition Seminar
22 September 2020
Problem: commits introducing inconsistencies

/**
 * Checks if one of the graphs is from unsupported graph type and throws
 * IllegalArgumentException if it is. The current unsupported types are
 * graphs with multiple-edges.
 *
 * @param graph1
 * @param graph2
 *
 * @throws IllegalArgumentException
 */

protected static void assertUnsupportedGraphTypes(
    Graph g1,
    Graph g2)
    throws IllegalArgumentException
{
...}
Problem: commits introducing inconsistencies

/**
 * Checks if one of the graphs is from unsupported graph type and throws
 * IllegalArgumentException if it is. The current unsupported types are
 * graphs with multiple-edges.
 * @param graph1
 * @param graph2
 * @throws IllegalArgumentException
 */

protected static void assertUnsupportedGraphTypes(
- Graph g1,
- Graph g2)
+ Graph g)
    throws IllegalArgumentException
{...}
Problem: commits introducing inconsistencies

/**
 * Checks if one of the graphs is from unsupported graph type and throws
 * IllegalArgumentException if it is. The current unsupported types are
 * graphs with multiple-edges.
 * 
 * @param graph1
 * @param graph2
 * @param g
 *
 * @throws IllegalArgumentException
 */

protected static void assertUnsupportedGraphTypes(
    Graph g)
    throws IllegalArgumentException{
    ...}

March 2006

November 2013

... still inconsistent
7.5 years later
Solution idea: detect inconsistencies with NLP

**Step 0**: get the diff of the change

**Step 1**: extract *natural language cues* from both code and comments and store as Bag-of-Words (BoW)

**Step 2**: measure BoW similarities

- **code v.0**

- **code v.1**

- **comment**
  - `[1:illeg, 1:argument, 1:throw, 1:one, 1:except, 1:check, 1:unsupport, 1:type, 2:graph]`

**Word Mover’s Distance**

- **WMD similarity**
  - 70%
  - 66% (-4%)
Implementation prototype
Implementation details

Implemented

- **Parser**: processes Java source code

- **Mapper**:
  - produces bag-of-words representations for code and comments
  - relates them based on similarity measures: cosine similarity, word mover’s distance

Working on

- **Change extractor**: work with AST-based diffs for clear code snippet scopes
- **Change analyzer**: combine information from the mapping and from the diff
- **Fix advisor**: notify the programmer if the comment needs more changes
Preliminary evaluation of mapping accuracy

- get public inconsistencies dataset from ICPC’19
- take first 50 commits and filter out fixes not detected (yet) by upDoc
- analyze 67 fixes in 20 commits that remained after filtering

Results:
- For 50 fixes the similarity scores improve as expected
- For 10 fixes the similarity scores did not change
- For 7 fixes upDoc reports unexpected decreases in the similarity scores
Inconsistency fix, correctly detected:

- This automatically calls `@link #escapeHTML` so make it safe for HTML too.
+ This automatically calls `@link #sanitizeForHtml` so make it safe for HTML too.
  
  @param string
  @return the sanitized string or null (if the parameter was null).

```java
public static String sanitizeForJs(String str){...}
```

**new version has better similarity score than old version**
Evaluation highlights 2/3

Readability fix, correctly skipped:

- Invoked when a refresh of current IndexReaders is detected as necessary.
- The implementation is blocking to maximize reuse of a single IndexReader (better for buffer usage, caching, ..) and to avoid multiple threads to try and go opening the same resources at the same time.
- @return the refreshed IndexReader

+ Invoked when a refresh of current @code IndexReader}s is detected necessary.
++
++ The implementation is blocking to maximize reuse of a single @code IndexReader} (better for buffer usage, caching, ..) and to avoid multiple threads trying and opening the same resources at the same time.
++
++ @return the refreshed @code IndexReader}

*/
private synchronized IndexReader refreshReaders() {
Evaluation highlights 3/3

Inconsistency fix, **NOT** correctly detected:

* Fetch an `@link InputStream` for each `@link Resource` in this `@link ResourceList`, pass the
* `@link InputStream` to the given `@link InputStreamConsumer`, then close the `@link InputStream` after the
-- `@link InputStreamConsumer` returns.
++ `@link InputStreamConsumer` returns, by calling `@link Resource#close()`.

```java
public void forEachInputStreamThenClose(final InputStreamConsumer inputStreamConsumer,
   final boolean ignoreIOExceptions) {
```