Towards Detecting Inconsistent Comments in Java Source Code Automatically

Natalia Stulova, Arianna Blasi, Alessandra Gorla, Oscar Nierstrasz


27 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
{

    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 g
 * @throws IllegalArgumentException
 */
protected static void assertUnsupportedGraphTypes(
    Graph g)

    throws IllegalArgumentException
{...}
Detecting inconsistent comments on change with NLP
Detecting inconsistent comments on change with NLP
Detecting inconsistent comments on change with NLP
Detecting inconsistent comments on change with NLP

creates natural language representations of code snippet and its comment
Detecting inconsistent comments on change with NLP

- Mapper creates natural language representations of code snippet and its comment.

- Mapper relates the two representations based on a similarity measure: cosine/word mover’s distance.
Detecting inconsistent comments on change with NLP
Detecting inconsistent comments on change with NLP

- - -

works with AST-based diffs for clear code snippet scopes
Detecting inconsistent comments on change with NLP
Detecting inconsistent comments on change with NLP

upDoc

Parser

Mapper

Change Extractor

Change Analyzer

Fix Advisor

Report

Fix Suggestions

combines information from the mapping and from the diff
Detecting inconsistent comments on change with NLP
Detecting inconsistent comments on change with NLP

upDoc

Parser

Mapper

Change Extractor

Change Analyzer

Fix Advisor

Report

Fix Suggestions

notifies the programmer if the comment needs more changes
The core part: code-comment mapping

---

**Step 1**: extract natural language cues from both code and comments and store as a bag-of-words

**Step 2**: measure similarity between two bags-of-words, add a mapping link if it is high enough

---

**Comment**

\[
\{\text{check, one, graph, unsupport, type, throw, illeg, argument, except}\} \]

**Code v.0**

\[
\{\text{void, assert, unsupport, graph, type, throw, illeg, argument, except, first, second}\} \]

**Code v.1**

\[
\{\text{void, assert, unsupport, graph, type, throw, illeg, argument, except, g}\} \]

**WMD similarity**

- **Code v.0**: 70%
- **Code v.1**: 66% (-4%)
Preliminary evaluation of mapping accuracy

Data

- use public inconsistent documentation fixes dataset from ICPC’19 produced by REVEAL group from USI
- take first 50 commits and filter out commits with fixes upDoc does not detect (yet)
- analyze 67 fixes in 20 commits remaining

Hypothesis

upDoc would report higher similarity scores in a mapping for the fixed version
Preliminary evaluation of mapping accuracy

Results

- 50 fixes - similarity scores improve as expected
- 10 fixes - the similarity scores did not change (formatting fixes)
- 7 fixes - unexpected decreases in similarity scores (not supported by upDoc)

Hypothesis confirmed

**upDoc** would report higher similarity scores in a mapping for the fixed version
Towards Detecting Inconsistent Comments in Java Source Code Automatically

Further discussion:

Q1: there are many approaches for automatic comment generation, yet can we improve them with our technique to capture implementation faults earlier?

Q2: machine learning is used more and more in software engineering tasks, yet is it an alternative or more of a complement to our approach?