Code clones

Code clones are fragments of source code that are identical or similar.

Copy-paste programming is a common anti-pattern in software engineering.

```c
int array_a[], array_b[];
...

int sum_a = 0;
for (int i = 0; i < array_a.size(); i++)
    sum_a += array_a[i];
int average_a = sum_a / ;

int sum_b = 0;
for (int i = 0; i < array_b.size(); i++)
    sum_b += array_b[i];
int average_b = sum_b / 4;
```
**Code clone types [1/4]**

**Exact clones**: identical code fragments which may have some variations in whitespace, layout, and comments

```c
if (a >= b) {
    c = d + b;
    // Comment1
    d = d + 1;
} else
    c = d - a; //Comment2

if (a >= b) { // Comment1'
    c = d + b;
    d = d + 1;
} else // Comment2'
    c = d - a;
```
Code clone types [2/4]

Renamed/parametrized clones:
syntactically equivalent fragments
with some variations in
identifiers, literals, types,
whitespace, layout and comments

```java
if (a >= b) {
    c = d + b; // Comment1
    d = d + 1;
} else
    c = d - a; //Comment2

if (m >= n) { // Comment1'
    y = x + n;
    x = x + 5; //Comment3
} else
    y = x - m; //Comment2'
```
Near miss clones: syntactically similar code with inserted, deleted, or updated statements
Code clone types [4/4]

---

**Semantic clones:** semantically equivalent, but syntactically different code

\[ n! = n \cdot (n - 1) \cdot (n - 2) \cdot (n - 3) \cdots \cdot 3 \cdot 2 \cdot 1. \]

```c
// factorial of VALUE
int i, j=1;
for (i=1; i<=VALUE; i++)
    j=j*i;

int factorial(int n) {
    if (n == 0) return 1;
    else return n * factorial(n-1);
}
```
Comment clones VS code clones

- similar definition: fragments of source comments that are identical or similar
- comment clones are more critical for program comprehension than code clones, though both can lead to bugs introduction
- comment clones are harder to classify, as natural language structure is much less restricted
Comment clone detection

RepliComment is a tool that implements a heuristic-based approach to comment clone detection for Java.

It focuses on method-level comment clones within one source file (~single class in most cases).
Comment clone types [1/3]

Copy-paste comment clones: a comment is copied from a correctly documented method and pasted by mistake to another one, whose functionality differs completely.

```java
public class CharMatcher {

    /**
     * @return true if this matcher matches every character in the
     * sequence, including when the sequence is empty.
     */
    public boolean matchesAllOf(CharSequence sequence) {...}

    /**
     * @return true if this matcher matches every character in the
     * sequence, including when the sequence is empty.
     */
    public boolean matchesNoneOf(CharSequence sequence) {...}

```
Comment clone types [2/3]

---

**Poor information clones:** uninformative comments that are so generic that they can be used for different methods.

```java
private class UserGroupInformation {

    /**
     * @return true or false
     */
    @InterfaceAudience.Public
    @InterfaceStability.Evolving
    public synchronized static boolean isLoginKeytabBased() throws IOException {...}

    /**
     * @return true or false
     */
    public static boolean isLoginTicketBased() throws IOException {...}

}  
```

Apache Hadoop 2.6.5
Comment clone types [3/3]

Legitimate clones: comments for methods that offer same so very similar functionality (e.g., polymorphic methods, overriding, common parameters,...).

```java
public class SolrClient {

/**
 * Deletes a single document by unique ID
 * @param collection the Solr collection to delete the document from
 * @param id the ID of the document to delete
 */
public UpdateResponse deleteById(String collection, String id) {...}

/**
 * Deletes a single document by unique ID
 * @param id the ID of the document to delete
 */
public UpdateResponse deleteById(String id) {...}

Apache solr 7.1.0
```
RepliComment clone detection and severity classification

Parser

<comment, signature> tuples

Clone detector

labeled clones .CSV

Clone analyzer

clone severity .TXT

<table>
<thead>
<tr>
<th>copy-paste</th>
<th>non-legitimate</th>
<th>high, mild, or low severity</th>
</tr>
</thead>
<tbody>
<tr>
<td>poor information</td>
<td>non-legitimate</td>
<td>high, mild, or low severity</td>
</tr>
<tr>
<td>false positives</td>
<td>legitimate</td>
<td>-</td>
</tr>
</tbody>
</table>

legitimate
non-legitimate
Filtering out legitimate clones

First, whole comment clones are never considered legitimate.

Second, comment parts are legitimate clones if they:

- are cloned for methods with same names
- document same exception type (in 4+ words)
- describe same parameter
- explain return value of the same non-primitive type
  (common in methods that update the class instance and return it, documented as “@return a reference to this”)
Comment clone severity analysis

signatures of two methods and the comment clone

\(<m_1, m_2, c>\)

- **c** is full comment
  - **Y**: \(m_1\) overloads **N**: \(m_2\)
    - **Y**: MILD; **N**: HIGH
      - **Y**: fix @m1; **N**: fix both
    - **Y**: MILD; **N**: fix both
      - **Y**: \(\text{sim}_1 < 0.25\); **N**: \(\text{sim}_2 < 0.25\)
        - **Y**: MILD; **N**: fix both
      - **Y**: \(\text{sim}_1 > 0.5\); **N**: \(\text{sim}_2 > 0.5\)
        - **Y**: Low; **N**: fix @maxSim
      - **Y**: \(|\text{sim}_1 - \text{sim}_2| > 0.1\)
        - **Y**: HIGH; **N**: fix both

similarity score: 

0 - 0.25 - 0.5 - 1
Method-comment similarity

---

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

\[
\text{comment} \{ [1:\text{check}, 1:\text{one}, 2:\text{graph}, 1:\text{unsupport}, 1:\text{type}, 1:\text{throw}, 1:\text{illeg}, 1:\text{argument}, 1:\text{except}] \}
\]

\[
\text{code} \, v.0 \{ [1:\text{void}, 1:\text{assert}, 1:\text{unsupport}, 4:\text{graph}, 1:\text{type}, 1:\text{throw}, 1:\text{illeg}, 1:\text{argument}, 1:\text{except}, 1:\text{first}, 1:\text{second}] \}
\]

\[
\text{code} \, v.1 \{ [1:\text{void}, 1:\text{assert}, 1:\text{unsupport}, 2:\text{graph}, 1:\text{type}, 1:\text{throw}, 1:\text{illeg}, 1:\text{argument}, 1:\text{except}, 1:\text{g},] \}
\]

Word Mover's Distance

\text{WMD similarity} 70\%

\text{WMD similarity} 66\% (-4\%)
Example analysis message

--- Record #53 file:2020_JavadocClones_log4j.csv ----
In class: org.apache.log4j.lf5.LogRecord
1) The comment you cloned:
   "(@return)The LogLevel of this record."
seems more related to <LogLevel getLevel()> than <Throwable getThrown()>
It is strongly advised to document method <Throwable getThrown()> with
a different, appropriate comment.

--- Record #152 file:2020_JavadocClones_hadoop-hdfs.csv ----
In class: org.apache.hadoop.hdfs.util.LightWeightLinkedSet
1) The comment you cloned:
   "(@return)first element"
seems more related to <T pollFirst()> than <List pollN(int n)>
It is strongly advised to document method <List pollN(int n)> with
a different, appropriate comment.
## Evaluation

<table>
<thead>
<tr>
<th>project</th>
<th>classes</th>
<th>LOC</th>
<th>GitHub</th>
</tr>
</thead>
<tbody>
<tr>
<td>elasticsearch-6.1.1</td>
<td>2906</td>
<td>300K</td>
<td>50K</td>
</tr>
<tr>
<td>hadoop-common-2.6.5</td>
<td>1450</td>
<td>180K</td>
<td>11K</td>
</tr>
<tr>
<td>vertx-core-3.5.0</td>
<td>461</td>
<td>48K</td>
<td>11K</td>
</tr>
<tr>
<td>spring-core-5.0.2</td>
<td>413</td>
<td>36K</td>
<td>38K</td>
</tr>
<tr>
<td>hadoop-hdfs-2.6.5</td>
<td>1319</td>
<td>262K</td>
<td>11K</td>
</tr>
<tr>
<td>log4j-1.2.17</td>
<td>213</td>
<td>21K</td>
<td>718</td>
</tr>
<tr>
<td>guava-19.0</td>
<td>469</td>
<td>70K</td>
<td>38K</td>
</tr>
<tr>
<td>rxjava-1.3.5</td>
<td>339</td>
<td>35K</td>
<td>43K</td>
</tr>
<tr>
<td>lucene-core-7.2.1</td>
<td>825</td>
<td>103K</td>
<td>4K</td>
</tr>
<tr>
<td>solr-7.1.0</td>
<td>501</td>
<td>50K</td>
<td>4K</td>
</tr>
</tbody>
</table>
Effectiveness of comment clone detection

**Total:** 11K non-legitimate and 61K legitimate comment clones

(non-legitimate clones by severity level)
Accuracy of comment clone classification

- False positives: we manually verify random 225 (~2%) samples of comment clone cases with HIGH, MILD and LOW severity level
  - ~ 21% (or 1 in 5 is a legitimate comment clone)

- False negatives: we manually verify 200 random samples of legitimate comment clones (20 for each project)
  - only 1 case in which we disagreed with the tool
Identifying clones in comments: summary

```java
public class CharMatcher {
    /**
     * @return true if this matcher matches every character in the
     * sequence, including when the sequence is empty.
     */
    public boolean matchesAllOf(CharSequence sequence) {...}

    /**
     * @return true if this matcher matches every character in the
     * sequence, including when the sequence is empty.
     */
    public boolean matchesNoneOf(CharSequence sequence) {...}
}
```

Total: 11K non-legitimate and
61K legitimate comment clones

(non-legitimate clones by severity level)