Fine-grained and Accurate Source Code Differencing

Problem Statement
- Existing approaches usually represent code changes or edit operations as add line or delete line actions
- Such representations are not precise
  - E.g., code move or update is not properly represented

Contributions
- GumTree—a novel efficient AST differencing algorithm that includes move actions
- An automated evaluation of GumTree
- A manual evaluation to compare GumTree vs. textual diff
- An automated evaluation to compare GumTree vs. ?

The GumTree Algorithm
- 1. A greedy top-down algorithm to find isomorphic sub-trees of decreasing height. Mappings are established between the nodes of these isomorphic subtrees. They are called anchors mappings.

The GumTree Algorithm (cont’d)
- 2. A bottom-up algorithm where two nodes match (called a container mapping) if their descendants (children of the nodes, and their children, and so on) include a large number of common anchors. When two nodes match, we finally apply an optimal algorithm to search for additional mappings (called recovery mappings) among their descendants.

The GumTree Algorithm (cont’d)
- 3. Recovery Mappings: to find additional mappings between leaf nodes and similar nodes
- 4. Generate edit operations for the unmatched nodes:
  - Insert
  - Delete
  - Update
  - Move
Top-Down Phase

- Start with the roots and check if they are isomorphic or identical. If not, the children nodes are then tested
- To identify the unchanged part
- Implementation
  - By hardcoding subtrees, the isomorphism test's complexity is $O(1)$
  - The worst-case complexity is $O(n^2)$

Bottom-Up Phase

- Search for container mappings, that are established when two nodes have a significant number of matching descendents

$$
dice(t_1, t_2, M) = \frac{2 | \{ t_1 \in s(t_1) | (t_1, t_2) \in M \} |}{|s(t_1)| + |s(t_2)|}
$$

Recovery Mappings

- Given two trees, find their additional mappings between the descendents,
  - remove the matched descendents, and
  - apply an optimized algorithm to find a shortest edit script without move actions

Architecture

- Parser: Java, JavaScript, R, and C
- Mappings: GumTree, ChangeDistiller, XYDiff, RTED
- Output: XML representation of AST, web-based view of an edit script, XML representation of an edit script

Evaluation

Comparison between GumTree, textual diff, and RTED
- The median of parsing time is 10
- Computing an edit script is only slightly slower than just parsing the files (median at 18 for Jenkins and 30 for JQuery)
Manual Evaluation

<table>
<thead>
<tr>
<th></th>
<th>Full (3/3)</th>
<th>Majority (2/3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GT does good job</td>
<td>122</td>
<td>137</td>
</tr>
<tr>
<td>GT does not good job</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Neutral</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ \phi 1 \]

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GT better</td>
<td>28</td>
<td>66</td>
</tr>
<tr>
<td>Diff better</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>Equivalent</td>
<td>45</td>
<td>61</td>
</tr>
</tbody>
</table>

\[ \phi 2 \]

Table 1: Agreements of the manual inspection of the 144 transactions by three raters for Question \#1 (top) and Question \#2 (bottom).

- **GumTree’s output is sometimes better than textual diff**

---

Automatic Evaluation

- **More matches = better**

### GDG

<table>
<thead>
<tr>
<th></th>
<th>GT better</th>
<th>CD better</th>
<th>Equal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mappings</td>
<td>3987 (31.5%)</td>
<td>342 (4.24%)</td>
<td>8242 (64.44%)</td>
</tr>
<tr>
<td>ES size</td>
<td>4598 (36.4%)</td>
<td>413 (3.22%)</td>
<td>7462 (54.16%)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>GT better</th>
<th>CD better</th>
<th>Equal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mappings</td>
<td>5976 (38.49%)</td>
<td>201 (1.59%)</td>
<td>4211 (29.92%)</td>
</tr>
<tr>
<td>ES size</td>
<td>6650 (66.97%)</td>
<td>175 (1.37%)</td>
<td>2250 (17.66%)</td>
</tr>
</tbody>
</table>

### JDTG

<table>
<thead>
<tr>
<th></th>
<th>GT better</th>
<th>RTED better</th>
<th>Equal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mappings</td>
<td>3908 (31.84%)</td>
<td>1204 (9.60%)</td>
<td>8752 (68.42%)</td>
</tr>
<tr>
<td>ES size</td>
<td>8920 (24.61%)</td>
<td>2193 (15.14%)</td>
<td>7727 (58.25%)</td>
</tr>
</tbody>
</table>

Table 2: Number of cases where GumTree is better (resp. worse and equivalent) than ChangeDistiller (top), middle) and RTED (bottom) for 2 metrics, number of mappings and edit script size (ES size), at the CDG granularity (top) and JDTG granularity (middle, bottom).

---

Automatic Evaluation (cont’d)

- **GumTree generates smaller edit scripts in most cases than RTED and ChangeDistiller**
  - 130 elements include move-only actions

### CD

<table>
<thead>
<tr>
<th></th>
<th>GT only move op</th>
<th>GT other op</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD only move op</td>
<td>77</td>
<td>1</td>
</tr>
<tr>
<td>CD other op</td>
<td>52</td>
<td>12862</td>
</tr>
</tbody>
</table>

Table 3: Comparison of the number of move operations from GumTree and ChangeDistiller for 12792 file pairs to be compared.