Evaluation of Code Clone Detection Techniques in the Context of Computer Science Education

Raj Koushik Reddy Korpally (rk2277@rit.edu)
Advisor: Prof. Carlos R. Rivero
Rochester Institute of Technology

Introduction
Code clones are a common phenomenon in computer science educational scenarios. Studies comparing the usefulness of cloning tools are limited. This project compares and evaluates the clone detection tools SourcererCC, NiCad, JPlag, ConQat, and Deckard.

Objectives
- Evaluate the cloning tools using BigCloneBench framework.
- Design a clone tool evaluation system based on CodeChef programs data.
- Study and compare the usefulness of results from both systems.

Background
Targeted types of code clones
- Type 1: Exact copy, only differences in whitespace and comments.
- Type 2: Variable renaming + any Type 1 differences.
- Type 3: Changing or adding few statements + any Type 2 differences.
- Type 4: Semantically identical codes, does not have same syntax.

Core algorithm used for clone detection by each tool
- SourcererCC: Tokenization and indexing comparison.
- NiCad: Largest Common Subsequence.
- ConQat: Abstract Syntax Tree, Vector matching.
- Deckard: Suffix Tree, Token comparison.

General Flow of Clone Detection Tools

Cloning Detection

Program 1:}
```java
public static int gcd(int x, int y)
{ // added comment
    while(x != y)
    { 
        if(x <= y) 
        { 
            y = y - x;
        } else
        { 
            x = x - y;
        }
    }
    return x;
}
```

Program 2:}
```java
public static int gcd(int x, int y) {
    // modified comment
    while(x != y) {
        if(x <= y) {
            y = y - x;
        } else {
            x = x - y;
        }
    }
    return x;
}
```

Program 3:}
```java
public static int gcd(int a, int b) {
    // semantically alternative implementation of gcd
    while(b != 0) {
        int k = a;
        a = b;
        b = k % b;
    }
    return a;
}
```

Results

<table>
<thead>
<tr>
<th>Tool</th>
<th>NiCad</th>
<th>SourcererCC</th>
<th>ConQat</th>
<th>Deckard</th>
<th>JPlag</th>
<th>SourceCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>9251 ms</td>
<td>20444 ms</td>
<td>82000 ms</td>
<td>135684 ms</td>
<td>42150 ms</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Running time on the CodeChef test data for clones identification

Conclusions
- Clone detection techniques are compared and evaluated by BigCloneBench and CodeChef based systems; the results justify the internal mechanisms used by tools to identify clones and the nature of reference clones compared.
- NiCad has the fastest performance for clones generation, NiCad and SourcererCC have most overlapping clones in the CodeChef system.
- Study recognizes the techniques failure to detect type 4 clones and the need for including tools identifying semantically similar clones as a future study.

References
(3) SourcererCC: Source of the UCD programs used in the clone detection example, https://www.gforge.rit.edu/trac/seg/wiki/projects/code

Evaluation System Design

Evaluation of Clones

Figure 1: Clone detection by Tokens comparison. Programs 1 and 2 match as clones (latest SourcererCC version).

Figure 2: Process for evaluation of tools using CodeChef data.

Figure 3: Recall values comparison between BigCloneBench and CodeChef Systems.

Tool
- Tool Deckard has the best recall value averaged over all categories but it is the slowest in clone detection and also has a large number of false positives.
- ConQat, JPlag perform poorly on BigCloneBench due to granularity mismatch between reference clones and detected clones.
- Overall recall values in CodeChef system are lower when compared to BigCloneBench evaluation.