Incremental clone detection for IDEs using dynamic suffix arrays

Jakob Konrad Hansen

University of Oslo

2023

Outline

Motivation

- Duplicated code is generally considered harmful to software quality
- Code clone detection, analysis and management is therefore important
- Incremental clone detection algorithms have not been thoroughly researched
- Incremental algorithms are useful in use-cases such as in IDEs

Our contribution

- CCDetect-LSP: An incremental clone detection tool for IDEs
- Uses a novel application of dynamic extended suffix arrays for clone detection
- Language- and IDE agnostic via Tree-sitter and LSP

Code clones

Definition (Code snippet)

A code snippet is a piece of contiguous source code in a larger software system.

Definition (Code clone)

A code clone is a code snippet which is equal or similar to another code snippet. The two code snippets are both code clones, and together they form a clone pair. Similarity is determined by some metric such as number of equal lines of code.

Clone types

- Code clones are classified into four types
  - Type-1: Syntactically identical
  - Type-2: Structurally identical
  - Type-3: Structurally similar
  - Type-4: Functionally similar (generally)

Clone type examples: type-1 and type-2

```
for (int i = 0; i < 10; i++) {
    print(i);
}
```

```
for (int i = 0; i < 10; i++) {
    // A comment

    print(i);
}
```

Figure: Type-1 clone pair

```
for (int i = 0; i < 10; i++) {
    print(i);
}
```

```
for (int j = 5; j < 20; j++) {
    print(j);
}
```

Figure: Type-2 clone pair

Clone type examples: type-3 and type-4

```
for (int i = 0; i < 10; i++) {
    print(i);
}
```

```
for (int i = 0; i < 10; i++) {
    print(i);
    print(i*2);
}
```

Figure: Type-3 clone pair

```
print((n*(n-1))/2)
```

```
int sum = 0;
for (int i = 0; i < n; i++) {
    for (int j = i+1; j < n; j++) {
        sum++;
    }
}
print(sum);
```

Figure: Type-4 clone pair

Clone detection

Clone matching techniques

- Text-based detection
  - Match based on raw source code
- Token-based detection
  - Match based on tokens
- Syntactic detection
  - Match based on AST
- Hybrid detection
  - Combine multiple approaches

Suffix array

Implementation: LSP architecture and functionality

- The Language Server Protocol (LSP) facilitates IDE agnostic tooling
- CCDetect-LSP is implemented as an LSP server

Implementation: Initial clone detection

Implementation: Incremental clone detection

Results

Discussion

Conclusion