Filtering Clones for Individual User Based on Machine Learning Analysis

Jiachen Yang, Keisuke Hotta, Yoshiki Higo, Hiroshi Igaki, Shinji Kusumoto
Graduate School of Information Science and Technology, Osaka University
1-5, Yamadaoka, Suita, Osaka, 565-0871, Japan
Email: {jc-yang,k-hotta,higo,igaki,kusumoto}@ist.osaka-u.jp

***Abstract*—Results from code clone detectors may contain plentiful useless code clones, and judging whether a code clone is useful varies from user to user based on different purposes of them. We are planing a system to study the judgment of each individual user by applying machine learning algorithms on code clones. We describe the reason why individual judgment should be respected and how in this paper.**

***Keywords*-filtering, classify, machine learning, code clone detector, judgment of user, token-based**

I. Introduction

Huge efforts have been made to detect identical or similar code fragments from source code of software, with these code fragments called as "code clones" or simply "clones". Clone Detection Tools(referred as CDT) usually generate a long list of clones from source code, among which a small portion of clones are helpful to user in improving software quality by applying refactoring or locating similar bugs, but the rest are not so "interesting".

Several methods have proposed to filter out those "uninteresting" clones such as metric-based method described in [1]. These methods are useful as they gathered a universal standard of judging whether a clone is "interesting", but they either require professional knowledges from user such as what all these metrics mean, or hard to fit to individual use case of users such as filtering only the code clones that can be applied extracting-method refactoring on. Tairas and Gray proposed a classification method in [2]. Compared with our method, they are clustering by identifier names rather than structure of the identical clone fragments.

According to a survey we conducted, users of CDTs tend to classify clones differently based on their individual use cases, purposes or experience about code clones. With this observation, we are working on the idea of studying judgments of each user on clones, which results a new clone classifying method, entitled as Fica, Filter for Individual user on code Clone Analysis.

II. Motivating Example

We conducted a survey on several students[1], providing them 105 clone sets from result of CDT on source codes in C language detected from `bash-4.2`, asking them whether

```
wcstr = 0;
slen = mbstowcs (wcstr, s, 0);
if (slen == -1)
  slen = 0;
wcstr = (wchar_t *)xmalloc (sizeof (wchar_t) ...
mbstowcs (wcstr, s, slen + 1);
wclen = wcswidth (wcstr, slen);
free (wcstr);
return ((int)wclen);
```

(a) execute_cmd.c

```
if (wcharlist == 0)
{
  size_t len;
  len = mbstowcs (wcharlist, charlist, 0);
  if (len == -1)
    len = 0;
  wcharlist = (wchar_t *)xmalloc (sizeof (wchar_t) ...
  mbstowcs (wcharlist, charlist, len + 1);
}
```

(b) subst.c

Figure 1. Example of source code in bash-4.2

they are willing to perform refactoring. Table I shows a part of the result. In this table, a code clone set is marked as `O` if a student is willing to refactor it or as `X` if not so. We can see from this table that their attitudes toward these clones vary from person to person.

As an example, source of clone with ID 5 is showed in Figure 1. Because their functions are identical, S and U thought they could be merge together. But Y and M considered the fact that Figure 1b is a code fragment in a larger function which is more than 100 LOC therefore may be difficult to be refactored.

Also from Table I we can see that Y was more strict than other three students. In the comment to this survey he mentioned that only clones that contains an entire body of C function are candidates for refactoring. This unique standard

Table I
Survey of clones in bash-4.2

Clone ID	Y	S	M	U
1	X	O	O	O
2	X	X	O	O
3	O	X	X	O
4	X	O	X	O
5	X	O	X	O
Count of O	5	24	23	25
Count of X	100	81	82	80

[1]All of students are from Graduate School of Information Science and Technology, Osaka University

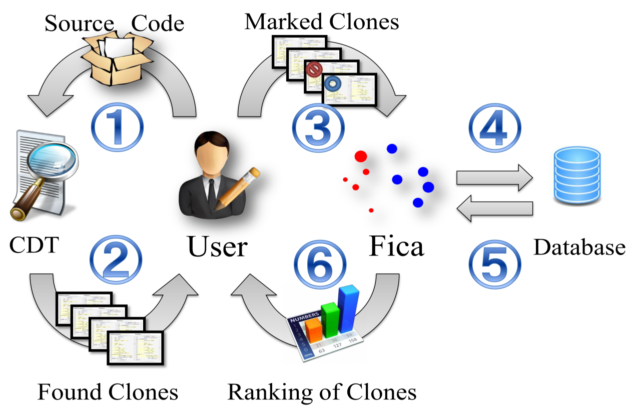

Figure 2. Overall Workflow of Fica with CDT

was also reflected in all 5 "interesting" clones he has chosen.

III. Workflow

In this section, we propose a workflow of Fica, which ranks detected clones based on studying historical behavior of a particular user, as a complement to existing CDTs that filtering unexpected clones.

A. *Overall Workflow*

The overall workflow of Fica is described in Figure 2.

1) User submits source code to a CDT.
2) CDT detects a set of clones in the source code.
3) User marks some of these clones as "interesting" or not according to her/his own judgment, and then submits these marked clones to Fica as a profile.
4) Fica records marked clones into its own database.
5) Meanwhile Fica studies characteristics of all these marked clones by using machine learning algorithms.
6) Fica ranks other clones remaining unmarked based on the result of machine learning, predicting the possibility of whether they are "interesting" or not to user.
7) User can further adjust the marks on code clones and re-submit them to Fica to obtain a better prediction. Fica will also record these patterns that it have learned into a database associated with the particular user profile so that further predictions can be made based on history decisions.

B. *Marking Clones Manually*

User of Fica is required to firstly mark a small set of clones found by CDT manually, to be used by Fica as an initial training set. Considered types of marks on clones can be boolean or tags:

bool	Clones are marked as "interesting" or not.
tags	Clones are marked as one of several tags or categories by user based on their use cases such as refactoring procedural, issue tracking id, etc.

As the most simple case, users need to tell Fica that they are interested in certain clones, and wants to find more like these. Tags typed marks can be considered as possible extension of boolean typed ones that involve multiple choices.

Also user of Fica is allowed to have multiple profiles in system, with each profile representing a use case of code clones. Profiles should be trained separately and Fica will treat them as individual users.

C. *Machine Learning*

Fica receives the clones from CDT and marks from user, studies the characteristic of the marked clones by calculating similarity of token sequence of these clones. This step employs machine learning algorithms which are widely used in natural language processing or text mining. The algorithm to be used will be similar with the one GMail used in detecting spam emails. By comparing the similarity of marked clones and unmarked ones, Fica can thus predict the possibility whether an unmarked clone is interesting or not to user profile.

IV. Conclusion and Future Plans

We have shown the fact that users of CDT may have different opinions on whether a code clone is "useful" or "interesting" to them. This observation suggested that filter of code clones should as well take user judgments into consideration to generate more useful list of code clones.

We are currently building the described system Fica, as a web-based system for proof of concept research purpose. The system consists of a generalized suffix tree [3] based CDT and a web-based user interface that allows the user marks detected code clones and shows ranked result.

We plan to release this system to public domain as soon as the system is ready. Collected user data and calculated result will also be released to public domain, which should be useful for further studies on clone classifying.

Acknowledgment

This study has been supported in part by Grants-in-Aid for Scientific Research (A) (21240002) and Grant-in-Aid for Exploratory Research (23650014) from the Japan Society for the Promotion of Science.

References

[1] Y. Higo, S. Kusumoto, and K. Inoue, "A metric-based approach to identifying refactoring opportunities for merging code clones in a java software system," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 20, no. 6, pp. 435–461, 2008.

[2] R. Tairas and J. Gray, "An information retrieval process to aid in the analysis of code clones," *Empirical Software Engineering*, vol. 14, pp. 33–56, 2009, 10.1007/s10664-008-9089-1.

[3] E. Ukkonen, "On-line construction of suffix trees," *Algorithmica*, vol. 14, no. 3, pp. 249–260, 1995.