I. Evaluation

In the process of finding test data, there is a benchmark of Detection of Software Clones: http://www.bauhaus-stuttgart.de/clones/. It is a general repository and information center for Detection of Software Clones, it accepts files with labeled clones for clone detection tool evaluation. However its format does not meet our requirement, so again we use manual selection to find "ground truth" for testing, just as we did in finding training data: using a low threshold to rule out most of the pairs and manually compare the rest.

We select test data from SWT–since machine learning is based on SWT files. The 10 files we used for testing are: *Button.java, CoolBar.java, Menu.java, Spinner.java, TabFolder.java, TableItem.java, ToolBar.java, ToolItem.java, Tracker.java, Tree.java*. The number of methods ranges from 38 to 147.

Since we have our tool and test data, we are ready to do evaluation. Table shows the test result:

A TABLE GOES HERE

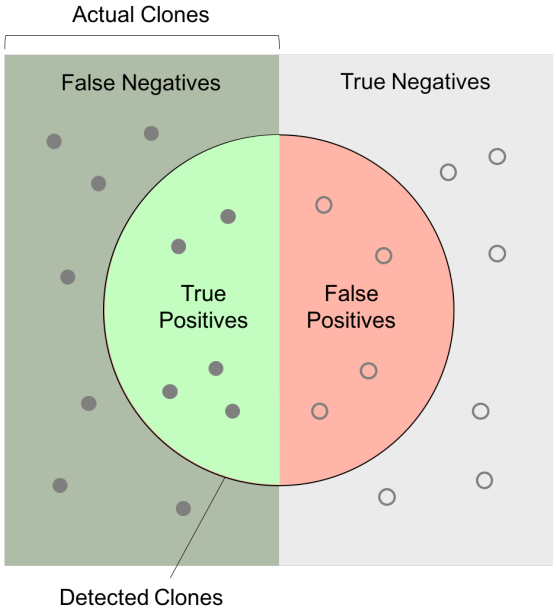

Fig. 1. Calculation of Precision and Recall

From Fig:1, based on the definition of Precision and Recall,

$$\text{Precision} = \frac{\text{True Positives}}{\text{Detected Clones}} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{Actual Clones}} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

we are able to evaluate our tool, the precision and recall is given in Fig:2.

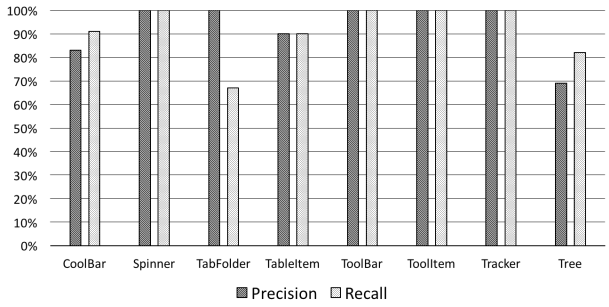

Fig. 2. Precision and Recall of STCD

The time cost of STCD is also evaluated, and is given in Fig:3. Since the comparasion for n methods is $\frac{n(n-1)}{2}$, we expect the time cost to be quadratic, and it turns out to be so–with some variations, which is the result of different method lengths.

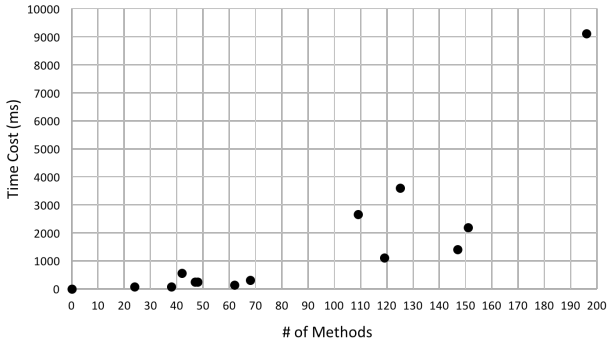

Fig. 3. Time Cost of STCD