I. Validity

Our approach has a high efficiency in finding clones, but there are still a number of threats to the validity:

1) Statistical limitation. STCD is a statiatical based tool, so each method needs to be big enough to get statistical data. In the actual testing, for methods less than 8 lines, the comparision is not reasonable, non-similar pairs can be taken as clones. So during the implementation of STCD, we ignored methods with less than 8 lines to reduce the false positives.
2) Data set too small. Both training data and test data set are small, we selected only 10 files in training and 10 files in testing, with actual clones no more than 20 in each file.
3) Data is chosen only from SWT. Both training data and test data are chosen from SWT, the good part of doing so is that machine learning can learn the developer's behavior, however on the other hand, it may not work so well for other projects. Training with user-defined files may help, but we don't know if the test result is as good as in SWT.
4) False negatives caused by manual selection. As discussed in the test results, by ruling out most unrelated comparisions with a low threshold, we got 2 actual clones ruled out, i.e. false negatives. It is unsure if there are more false negatives like this, so our result may differ from the truth.
5) Ambiguous match. Ambiguous match is realized by threshold. First is the shreshold of 0.7 used in the pretreatment of variable names. If the variable names are more similar than 0.7, they are considered the same and compared. However this may not be true. By a threshold of 0.7 we may either put unrelated variables together, or take similar variables as different.
   Second is the final threshold of 0.87. A lower threshod will result in a lower precision while a higher threshold will result in a low recall. But that is not absolute. It may either rule out actual clones or print out unrelated pairs.
6) Extra time caused by ASTParser Tool. In the first step of implementation, method declaration, we used ASTParser Tool to divide the file into methods and get the information for each method. ASTParser Tool is based on Abstract Syntax Trees, by transforming the original file into trees, extra time cost is introduced, which is not needed. A simpler method can be used to reduce this part of time.