EUROPEAN ACADEMIC RESEARCH
Vol. IV, Issue 2/ May 2016

Impact Factor: 3.4546 (UIF)
DRJI Value: 5.9 (B+)

ISSN 2286-4822
www.euacademic.org

Clone code detector using Boyer–Moore string search algorithm integrated with ontology editor

SYED MOHD FAZALUL HAQUE
Maulana Azad National Urdu University
fazal.manuu@gmail.com
V SRIKANTH
K L University
srikanth_vemuru@yahoo.com
E. SREENIVASA REDDY
Acharya Nagarjuna University
esreddy67@gmail.com

Abstract:

Program is defined as the set of instructions to perform a particular task. To write a program or to complete the task developer has to take care about code. Some developers will copy the code for some of the difficult tasks this is called code copy or cloning code. The process of cloning can lead to other issues of software development. The other software tools that are similar to code cloning are plagiarism and software tools (Roy and Cordy, 2007) etc. To do the better software maintenance we need to detect the duplicate code (baker, 1995). The duplication of code is happening frequently in web applications as well as windows applications. To overcome this problem the proposed boyer moore string search algorithm is integrated with code ontology editor is developed for better performance. The results will show the better performance to detect the cloned code.

Key words: Clone code, plagiarism, ontology, editor.

INTRODUCTION:

In programming languages, duplicate code is very difficult to find out from where the source code is copied from. The detection of similar source code files according to the methods, fields, properties etc. These source code files are from various student assignments and from various projects. It is very difficult to find out the duplicate code from big projects like ERP, Accounts package etc. In most of the applications duplicate is generated according to their requirement programmers are copying the code from various sources using internet. In this paper, our proposed system works on string search or method search algorithms according to the methods used in the program and parameters used in the method.

In project developments, the code reuse and component reuse is also the important task according to the functionality of the program the reuse the by the developers. The project is product based or client based may chance of duplicate code. Several researches says that approx 25% - 35% of large projects having the cloned codes (Krinke,2001). For some of the large projects detection of cloned code is flexible only by automatic techniques. There are several automatic techniques proposed to detect clones automatically (Bellon et al, 2007).

RELATED WORK:

No of works have been developed to detect the similarity codes by representing the graph or tree and also string based detection and semantic based detection. Code cloning or copying a code method for reusability by copying exactly the existed code without any modifications is known as code smell in software maintenance. This type of reusability of programming methods of existing code is called dup or clone code. Some people believe that the major cause cloning is by the act of copying and pasting of the code. Recently some of the

developers copy the templates from the internet and changing the background color, images and logos. This is happing in web applications developed in HTML, jsp, asp.net or php.

Clone is one that appears to be a copy of original form (Koschke, 2006). Baxter is his research work (Baxter et al, 2008), stated that a clone is program fragment that is identical to another fragment.

Krinke (Krinke, 2001) used the term "Similar code", Ducasse (Ducasse et al, 1999 instead of "Duplicate Code". Komondoor and horwitz, 2001 used the term "duplicate code" instead of "clone" as an instance of duplicate code. mayrand and leblanc in 1996 used metrics to find "an exact copy or a mutant of another function in the system.

PROPOSED SYSTEM:

In this paper, the proposed system works on Boyer-Moore string search algorithm for detection of duplicate code. To improve the performance of the string search algorithm an integrated ontology editor with common language compiler is developed for supporting various programming languages. Integrated ontology editor with common language compiler is used to focus the relation between functions, methods and statements used in the program.

BOYER-MOORE ALGORITHM:

- Comparison starts from right to left.
- It is very fastest algorithm to find out one string of charters in another by method oriented duplicate search.
- By remembering more of what has already been matched, one way to done larger skips through the text. We can even arrange ``perfect memory'' and thus look at each character once, whereas the Boyer-Moore algorithm linearly searches a character from multiple text by

multiple times. The concept something that one has undertaken by others. It suffers the need for very large tables or state machines.

- This algorithm is more efficient than previous which is compatible to our proposed code ontology editor.
- The algorithm scans the characters of the pattern from right to left beginning with the rightmost character. During the testing of a possible placement of pattern Add(int a,int b) against search text sum(int a, int b) a mismatch of text character T[i] = c with the corresponding pattern character P[j] is handled as follows: If c is not contained anywhere in P, then shift the pattern P completely past T[i]. Otherwise, shift P until an occurrence of the character c in P gets aligned with T[i].
- This technique likely to avoid lots of needless comparisons by significantly shifting pattern relative to text.
- A=actual method, B=user method
- A=Add[X];
- B=Sum[Y];
- If B is not contained anywhere in A, then shift pattern A completely past B. Shift A until an occurrence of charter a in A in A gets aligned with B. This technique may avoid lots of iterations by using shifting relations to the text.

METHOD MATCHING:

For every file pair, the Boyer Moore string search algorithm counts the number of matching words that are not keywords. To determine whether a word is a programming-language keyword, comparison is perform with a list of programming-language keywords. For example, the word if loop in a C#

source code file would be ignored as a keyword by this algorithm. In some programming languages such as C# and Java, keywords are case sensitive. In other programming languages like Visual Basic, keywords are not case sensitive. Boyer Moore string search algorithm has a switch to turn case sensitivity on/off depending on the language to be examined for a case-sensitive language like C, the word if loop is not be considered as a language keyword and would not be ignored. In case of insensitive language such as Visual Basic, however, the word if loop would be considered a language keyword and would be ignored. In either case, when comparing non keyword words in the file pairs, case is ignored so that the word Index in one file would be matched with the word index in the other. This case-insensitive comparison is done to prevent being fooled by simple case changes in plagiarized code.

BOYER MOORE MATCHER (SUM, ADD)

Input: Text with a characters and Pattern with b characters

Output: Index of the first substring of Sum matching Add

1. Compute function last
2. x ← b-1
3. y ← b-1
4. Repeat
5. If Add[y] = Sum[x] then
6. if y=0 then
7. return i // we have a match
8. else
9. x ← x -1
10. y ← y -1
11. else
12. x = x +b - Min(y, 1 + last[Sum[x]])
13. y = b -1
14. until x > n -1

15. Return "no match"

EXPERIMENTAL RESULTS:

The development of our integrated with code ontology editor using Boyer-Moore algorithm is developed with C#.net. C#.net is most featured language among all the other languages. With this one we are integrated with Common compiler for the compilation of the source code. As we seen in many of the cloned code tools here also we are providing ontology text editor for original code other text editor for pattern matching and word matching. Our tool supports for 32-bit processor and 320 gb hard disk. Our algorithm works very fast on every large method. For Example: Sample Addition of two numbers in C#.Net. Accuracy of our project is above 80% according to the matching content. Our tool supports HTML, ASP.NET, VB.NET, C, C++ and C#.NET. In the below source code we have sample code given in the input. Second program is the output.

This is input:

```
using System; namespace CsharpPrograms
{
class Program
{
static void Main(string[] args)
{
int x; int y; int result;
Console.Write("\n Enter the first number to be added: ");
x=Convert.ToInt32(Console.ReadLine());
Console.Write("\n Enter the second number
to be added: ");
y = Convert.ToInt32(Console.ReadLine());
result = x + y;
Console.Write("\n The sum of two numbers is: "+result);
Console.ReadLine();
```

```
}
}
}
```

Output for Cloned code (red text represent the Cloned code):

```
using System;
namespace Addition
{
class Program
        {
        static void Main(string[] args)
        {
                int a;
                int b;
                int c;

                Console.Write("\n Enter the first number to be added: ");
                a=Convert.ToInt32(Console.ReadLine());
                Console.Write("\n Enter the second number to be added: ");
                b = Convert.ToInt32(Console.ReadLine()); c = a + b; //Matched content according to the functionality
                Console.Write("\n The sum of two numbers is: "+c);
                Console.ReadLine();
}
}
}
```

CONCLUSION:

In this paper, the proposed duplication of code is happening frequently in web applications as well as windows applications.

To overcome this problem the proposed boyer moore string search algorithm is integrated with code ontology editor is developed for better performance. The results will show the better performance to detect the cloned code. In this paper, integrated code ontology editor is also implemented in this tool. This improved editor will provide better results.

REFERENCES

[1] AHO, A.V., 1990, Algorithms for finding patterns in strings: Handbook of Theoretical Computer Science, Volume A, Algorithms and complexity: J. van Leeuwen ed., Chapter 5, pp 255-300, Elsevier, Amsterdam.
[2] AOE, J.-I., 1994, Computer algorithms: string pattern matching strategies, IEEE Computer Society Press.
[3] BAASE, S., VAN GELDER, A., 1999, Computer Algorithms: Introduction to Design and Analysis, 3rd Edition, Chapter 11, pp. ??-??, Addison-Wesley Publishing Company.
[4]BAEZA-YATES R., NAVARRO G., RIBEIRO-NETO B., 1999, Indexing and Searching, in Modern Information Retrieval, Chapter 8, pp 191-228, Addison-Wesley.
[5]BEAUQUIER, D., BERSTEL, J., CHRÉTIENNE, P., 1992, Éléments d'algorithmique, Chapter 10, pp 337-377, Masson, Paris.
[6]BOYER R.S., MOORE J.S., 1977, A fast string searching algorithm. Communications of the ACM. 20:762-772.
[7]COLE, R., 1994, Tight bounds on the complexity of the Boyer-Moore pattern matching algorithm, SIAM Journal on Computing 23(5):1075-1091.
[8]CORMEN, T.H., LEISERSON, C.E., RIVEST, R.L., 1990. Introduction to Algorithms, Chapter 34, pp 853-885, MIT Press.
[9]CROCHEMORE, M., 1997. Off-line serial exact string searching, in Pattern Matching Algorithms, ed. A. Apostolico and Z. Galil, Chapter 1, pp 1-53, Oxford University Press.

[10] CROCHEMORE, M., HANCART, C., 1999, Pattern Matching in Strings, in Algorithms and Theory of Computation Handbook, M.J. Atallah ed., Chapter 11, pp 11-1--11-28, CRC Press Inc., Boca Raton, FL.