PLAGIARISM REVEALING IN THE SOURCE PROGRAM CODE ON THE BASIS OF ITS SYNTACTIC REPRESENTATION

Tetyana Shatovska, Iryna Kamenieva

Abstract*: This paper considers the existing approaches and models of data representation applicable for the problem task of plagiarism revealing in the source code. This paper present an approach based on the solution concerning syntactic comparison of source codes for the task of plagiarism revealing in students course code. The first phase of analysis, this calculation of quantitative characteristics of the source code, in other words, an attribute algorithm. This preliminary calculation allows us to obtain an overall picture of project files, which is subjected to analysis. Attribute algorithm allows us to discard obviously not like a pair of source files in a relatively short period of time, resulting in we will get the abbreviated set of couples who want to compare the structural algorithm. The next phase is he analysis of potentially similar files of a structural algorithm. In this paper, a measure of similarity of these files have been selected as the value of the greatest common subtree. This problem has much in common with the problem of isomorphism of graphs.*

Keywords*: Plagiarism, source code, isomorphic graph, detector of plagiarism.*

ACM Classification Keywords*: D.2.5 - Testing and Debugging*

Introduction

With rapid development of mediums and forms of communications on the basis of modern information technologies such problem, as plagiarism expands also. It widespread almost in every fields of human activity. This fact not only breaks the copyrights of the person, whose work has been used as plagiarism, but also completely destroys the essence of education itself. In the sphere of education this problematic has global scales.

Education is a process of the direct interaction of the student and the lecturer, where the student makes efforts to obtain knowledge, and the lecturer in his turn helps to master the obtained knowledge, and as result, to estimate them adequately. Student's work containing the plagiarism cannot be estimated adequately in the main for the obvious reasons. In many cases the lecturer is not able to recognize the plagiarism and accepts the work as fulfilled, estimating it with a high mark and does not assume, that this student has appropriated the results of another's work.

As is known, students of technical universities in IT sphere practice the program implementation of algorithms, information systems and other programs within the course. The main objective of the tasks performing is to give knowledge and to increase the comprehension in the specific area. Unfortunately, many students resort to plagiarism usage in their works. For example, one of the students has taken a source code written by other student and has claimed it as his own or has downloaded a source code by means of the Internet resource.

In the modern world the Internet is called-for and keep on gaining the popularity among the students and consequently the access to various resources increases.

Since for today the problem is rather actual, for its solution there are already various automated approaches and resources which are successfully applied to search the plagiarism in essays, notes to the yearly projects, diplomas and other text documents. The great number of tools in the Internet allows to verify the text for plagiarism, but thus the plagiarism revealing in a program code is not studied thoroughly. The plagiarism revealing in the source code has almost nothing common with plagiarism revealing in the text as programs among themselves can have the identical syntactic representation, but thus to have the various text. We reach

the conclusion, that for the solution of the set problem the perfect automated resource which will allow to speed up the plagiarism revealing process is necessary. Further in paper the matters concerning the automated revealing of plagiarism in the source code will be considered.

Plagiarism in the source code

Before starting to look into the various ways of plagiarism revealing it is necessary to determine the understanding of this term. To define the quality of plagiarism searching this term should be describe. The plagiarism searching task as well as definition itself has wide enough and not accurate sense, depending on a subject field it can be perceived in different ways. The term plagiarism has various related statements:

- Plagiarism – Kind of copyrights violation, consists in illegal usage under someone own name of another's work (scientific, literary, musical) or invention, the innovation proposal (completely or partially) without indicating the source of adoption [1]. Compulsion to the co-authorship is also considered as plagiarism [2].
- Plagiarism – literal adoption from another's literary work without indicating the source [3].
- Plagiarism (from Lat. plagio – I abduct) – kind of author or the inventor rights violation. Consists in illegal usage under someone own name of another's work (scientific, literary, musical) or inventions, the innovation proposal (completely or partially) without indicating the source of adoption. [4]
- Plagiarism – appropriation of fruits of another's creativity: publication of another's works under someone own name without indicating the source or usage without the reformative creative changes made by the borrower. [5]
- Plagiarism – deliberate appropriation of authorship to another's work of science, literature or art. It is not considered as plagiarism the adoption of a theme or a story of work or the scientific ideas composing its content, without adopting the form of their expression. [6]

The above mentioned definitions are accurate enough and close to the essence but, nevertheless, they are not considered the specificity of plagiarism in the program code, therefore to put them into practice as the basic definitions of plagiarism is difficult. To understand which features should be considered for plagiarism searching in the program code we will observe, what is the computer program, as well as source code.

- Program (from Gr. programma – the declaration, the instruction, the decree) – the ordered sequence of operations for the computer, realizing the algorithm of some task solution [4].
- Source code (also the source text) is a text of a computer program in any programming language. In the generalized sense – any input information for the compiler. The source code is either complied into executed code by means of the compiling program, or executed directly according to the text by means of the interpretive program [7].

Usually for the analysis of programs the source code containing the data required for the analysis is used. The source code is always linked to the programming language, namely with its grammar, therefore it is the structured text which has limitations set by grammar of the programming language. For this reason two programs having absolutely different text idea could be identical in syntactic representation, and, therefore, could be plagiarism. Let us consider the following instance:

```
public class A {
    private int a = 0;

    public int method(int b) {
        return a * b; }}
```

```
public class B {
    private int c = 0;

    public int anotherMethod(int d) {
        return c * d;}}
```

In this instance the elementary way of plagiarism hiding in the source code has been shown. It is obvious, that the source texts of the presented extracts do not coincide, but thus have absolutely identical logic and semantics. More detail the methods of plagiarism masking will be considered in the following sections of this paper.

The plagiarism searching in the program code demands the special approaches independent on text representation of the program, and also are steady to various ways of plagiarism masking.

Further we will consider the various models and approaches for plagiarism revealing.

Plagiarism searching in the source code

From the previous sections it becomes clear, that plagiarism searching in the source code cannot be realized by means of simple line-by-line comparison or by finding of the greatest common line, the plagiarism searching in the program code is complicated by that frequently they try to hide it. Now the following ways of plagiarism masking [8] are known:

- Code comments changing,
- Code formatting changing,
- Identifiers renaming,
- Change of operands order which are independent on execution time,
- Original data types substitution for related,
- Expressions substitution for identical in syntax,
- Operations and variables adding which are not in use or excessive,
- Code extracting from methods and on the contrary
- Original code combining with plagiarism
- Translation of the source code into other programming language

There are two basic approaches for plagiarism searching in the source code. First is based on program significant attributes counting, and the second is on structural representation of the source code.

The attribute method [8,9] is applied to digital expression of program significant attributes using further comparison of the obtained results with other analyzed programs. This method is capable to detect plagiarism in the masked source code. The significant attributes sampling depends on the programming language in which the source code is written. Usually using such approach the following basic attributes are considered:

- quantity of classes,
- quantity of methods,
- quantity of variables,
- quantity of loops,
- quantity of methods invocations,
- quantity of conditions,

The specifying attributes which are dependent on the programming language (quantity of abstracts, imports, enumerations and other specific features of the programming language) can be optionally used. In this approach it is very important to choose the correct metrics for further calculations to obtain proper result. Let us consider the instance showing the attribute method usage.

```
public class A {
    public int getSum(int[] array) {
        int sum = 0;
        for (int i = 0; i < array.length; i++) {
          sum += array[i];
        }
        return sum; }}
```

```
public class B {
    public int addition(int[] array) {
        int sum = 0;
        int i = 0;
        if (true) {
          while (i < array.length) {
                    sum += array[i++];
          }
        }
        return sum; }}
```

From the resulted above code it is obvious, that both classes considerably differ from each other in their text representation. Class A contains 1 method, 1 loop and 3 declarations of a variable, including arguments of

method and a variable declared in the loop. Let us consider class B as potential plagiarism of A class. Class containing plagiarism is masked by operand substitution for identical in syntax and also by adding of a superfluous operand which does not change logic of code execution and contains 1 method, 1 loop, 1 condition and 3 declarations of a variable including input arguments. As a result of such calculation it is possible to say, that programs are very similar and potentially are plagiarism.

The main principle of such approach operation has been shown in this instance. In practice it may be complicated by adding of weights for different metrics etc. Such approach potentially can give the correct result, but it is also ineffective enough, as two absolutely different programs which differs both text and syntactic representations can give similar result that is not acceptable to solve our problem.

Next and most prospective approach consists in comparison of programs taking into account their structure [8,9]. Such procedure is much more difficult, than simple detection of quantitative characteristics of the program. This is due to the complexity of the analysis of the source code structure. In such approach each characteristic of the program is analyzed not isolated, but in context of its parent element. All units of the program are linked hierarchically and are at various levels of the enclosure. The structural method exceeds the attribute one by quality, but can be worse by productivity as the analysis of the program structure could demand much more calculations, than during conventional counting of its quantitative characteristics.

The main principle of our method is the preliminary construction of the source code tree, namely Abstract Syntax Tree [10] including its further analysis. Having built the Abstract Syntax Tree we obtain the complete conception about the source code, its units, structure, interconnections that give the possibility to realize the algorithms which are dependent on these data. On the other hand such approach is not flexible enough for the reason, that each programming language requires the separate parser which builds the Abstract Syntax Tree. Each programming language can contain various units in the structure which are defined by grammar of language, that directly influences the structure of the program tree, correspondingly the analyzer will be linked to the certain type of the Abstract Syntax Tree. This approach adaptation to any of programming languages demands the considerable efforts. These deficiencies are the pay for the high-quality approach. The classical instance of this approach realization consists in comparison of subtrees of various programs structure.

To understand the efficiency of the structural approach let us consider the following instance:

```
public class A {
    private String concat(char separator, String... strings) {
        StringBuilder sb = new StringBuilder();
        for (String str : strings) {
          sb.append(str).append(separator); }
        return sb.toString();}}
```

```
public class B {
    class Inner {
        private String add(String[] values, char sep) {
          StringBuilder sb = new StringBuilder();
          int i = 0;
          while (i++ < values.length) {
                    sb.append(values[i]);
                    sb.append(sep); }
          return sb.toString();}}}
```

In the code of the program resulted above the class A is the original. Proceeding from it, it is obvious, that class B. Inner is plagiarism of the class A with insignificant alterations which do not influence the logic of program execution. For plagiarism masking the code has been wrapped in the internal class, the loop was substituted for identical one and the order of the method parameters declaration was changed. It is obvious, that text comparison of these source codes will not give demanded result. The text state of the program has been changed and differs from the original.

So, we need to build the structural representation of each of programs, namely Abstract Syntax Tree for each source code. Such procedure will allow to analyze the program from the point of view of its structure, having discard the superfluous information about the names of identifiers etc. Fig. 1 and 2 show the visualization of the tree structure of programs (original and plagiarism).

Original:

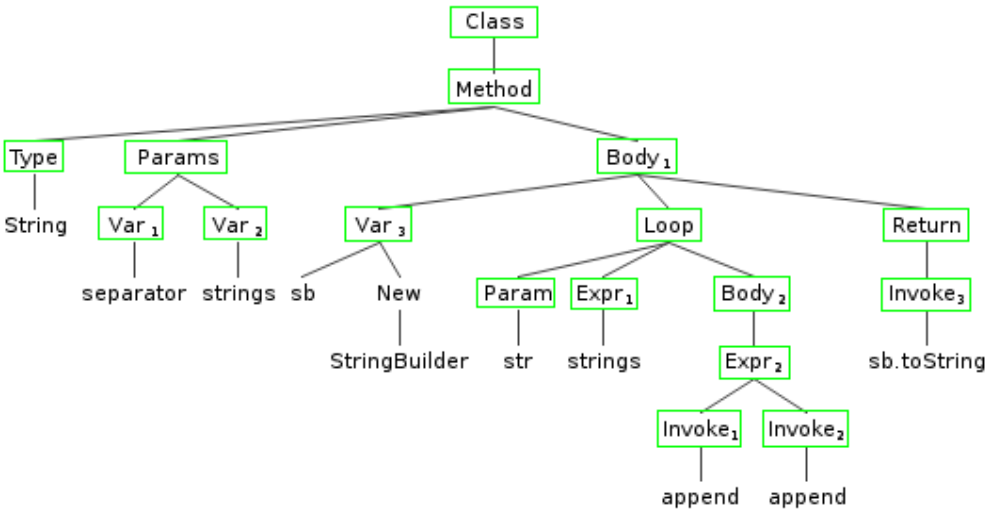

Figure 1 The original program tree structure visualization

Plagiarism:

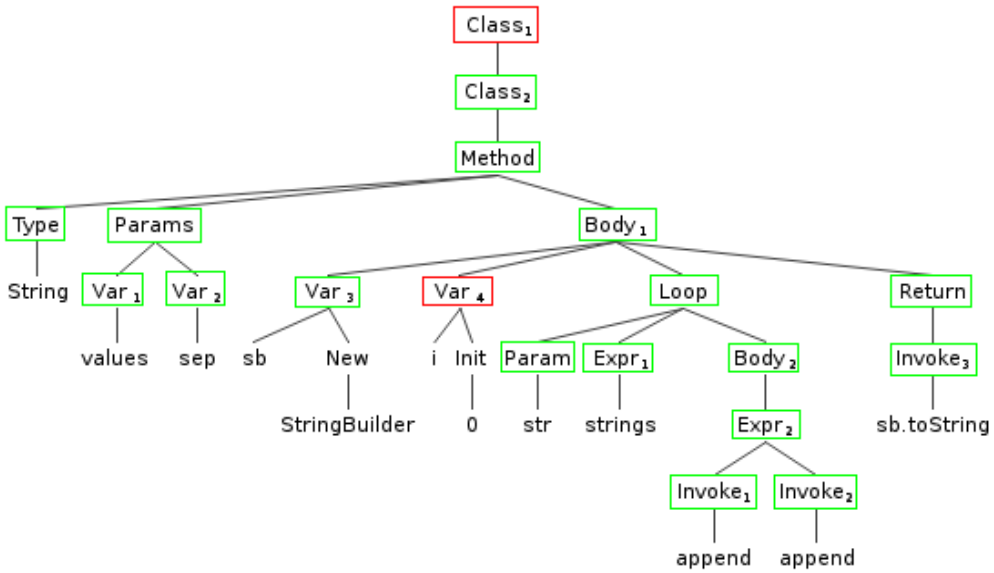

Figure 2 Plagiarism program tree structure visualization

Having analyzed both structures it is possible to conclude, that the structure of the original program is the subtree of the plagiarism program structure even taking into account such fact, that the code has been changed. Thus, we conclude, that program B is plagiarism. Programs can coincide not completely but only partially, small sections of one tree can coincide with small sections of the other tree, therefore, the common subtree can practically start with any unit. Proceeding from this fact, the complexity of any algorithm working under such principle increases, since it is necessary to search all units of the tree approximately so much times how many units in it. The algorithm complexity is cubic, that creates difficulties in practice usage because of low productivity.

It also should be noted, that there are approaches for plagiarism detection in the source code which can work on the basis of neural networks, classifications and clusterizations.

Combined approach

To obtain the high efficiency of algorithm the combined approach for plagiarism revealing in the source code is used. As it has been described above, the attribute method has enough productivity, but in its turn can give absolutely incorrect result, the structural method allows to reveal the plagiarism precisely enough, but demands labour-intensive calculations. Proceeding from it, we conclude that there is combining of these two approaches.
The small programs consisting of one file have been presented above, but in fact almost any program consists of the numerous files great number of which forms the program.
Any program has both big and small files with the source code, which are required to be analyzed for plagiarism revealing. These can be both simple matters and the big files with the source code where the basic functionality, algorithms etc. are realized
For algorithm improving the following approach is offered. The first phase of the analysis is the counting of quantitative characteristics of the source code (attribute algorithm). The preliminary calculation allows to obtain the common information about project files. The first stage does not require the difficult calculations and it means, that we obtain total characteristic about the source code without considerable expenses for productivity. It is known, that two files with the source code can give equal or very close quantitative characteristics and thus to have absolutely different logic of performance, therefore, we cannot be assumed only on the results obtained during the first phase. The attribute algorithm allows to discard certainly not similar pairs of files with the source code for short enough period. Therefore we will obtain the reduced ensemble of pairs which should be compared by means of structural algorithm.
The second phase - the analysis of the potentially similar files by the structural algorithm. In this paper the measure of similarity of such files the value of the greatest common subtree had been chosen. This problem has much in common with the problem of graphs isomorphism. Isomorphism is the common concept which is used in various sections of mathematics. Generally it can be described as follows: let two ensembles with certain structure (groups, rings, linear spaces, etc.) is given. The bijective mapping between them is called isomorphism if it saves this structure. Such ensembles with structure is called the isomorphic ones. Isomorphism always sets the relation of equivalence on the class of such ensemble with structure. The objects between which there is the isomorphism, are in a sense "equally arranged", they are called isomorphic ones. As classical instance of isomorphic systems can be ensemble R of all real numbers with the operation of addition defined on it and ensemble R + of the positive real numbers with the operation of multiplication set on it. Representation $x \rightarrow \exp(x)$ in this case is isomorphism.
During plagiarism identification in isomorphic graphs the main aim becomes the identification of largest common substructure. The size value of the common substructure, in our case it means subtree, determines the level of similarity between analyzed programs, the more value the more programs are similar.

Architecture of the application

Let us consider the common architecture of the offered detector of plagiarism, the components of which the application and units with which it will work should consist. Let us start from the description of the operation principle of the plagiarism detector. On the entry the set of files with the source code is feed, then these files should be present as a tree structure to be analyzed. As it has been told earlier, it is offered to use the combined approach to analyze the source code for plagiarism revealing. Therefore, in the considered architecture there is provided two phases of the analysis. The first and the fastest phase is based on counting of significant ensembles and allows to discard certainly not similar pairs, and the second one analyses the structure of potentially similar files. The system kernel could be divided at least into three basic components.

Figure 3 Main modules of the system

In the figure above you can see main modules of the system and the common sequence of performance. Let us consider the system components.

AST parser is required to construct Abstract Syntax Tree of the source code. This model of the data presentation is necessary for the analyzer of the programmed application. Parser is a front part of the application through which all input data pass. Then data are converted into required type, that is into AST. This part of the application is in any compiling program, compilation of any program begins from the construction of the source code tree.

Attribute Counting Algorithm is a model where the quantitative characteristics of the program are calculated by means of attribute algorithm on which entry the ensemble of the AST representations moves. As a result each tree is marked by the characteristics calculated for it and transferred to the next module.

Conclusions

On the basis of studies has been proposed and developed an hybrid approach to solve the plagiarism problem of the source code. The developed software tool has both scientific and practical value, hence the area of application. In terms of practices developed application can be used to search for plagiarism among a set of student projects with source code. In terms of scientific value, the result of the research, as well as evaluation of results can be used for further development of the developed approach.

Aknowledgement

The paper is published with financial support by the project ITHEA XXI of the Institute of Information Theories and Applications FOI ITHEA (www.ithea.org) and the Association of Developers and Users of Intelligent Systems ADUIS Ukraine (www.aduis.com.ua).

Bibliography

[Синельников et al, 1994] С. Синельников, Т. Соломоник, М. Биржаков. Энциклопедия предпринимателя – Санкт-Петербург: ОБИС, 1994.

[Сухарев, 2002] А.Я Сухарев. Большой юридический словарь – Москва: ИНФРА-М, 2002.

[Блокгауза et al, 2001] Ф.А. Блокгауза, И.А. Ефрона. Энциклопедический Словарь – 86 т., С.-Петербург: 1890 – 1907// http://slovari.yandex.ru/dict/brokminor/article/23/23998.html?text , 2001.

[Прохоров et al, 1978] Большая советская энциклопедия – 3-е издание: 30 томов, под ред. А.М. Прохоров и др. // http://bse.sci-lib.com, 1969 – 1978гг.

[Асмус et al, 1939] В. Ф. Асмус, Д. Д. Благой, Б. М. Гранде и др. Литературная энциклопедия под ред. В.М. Фриче, А.В. Луначарский и др. // http://slovari.yandex.ru/dict/litenc ,1929-1939 гг.

[http://www.glossary.ru/, 2005]Тематический толковый словарь // http://www.glossary.ru/ , 2005.

[Wikipedia] Wikipedia // http://ru.wikipedia.org/wiki/Исходный_код, см. 2010.

[Hamilton et al, 2008] J. Hamilton, S. Danicic. Static Source Code Analysis Tools and their Application to the Detection of Plagiarism in Java Programs // http://whoyouknow.co.uk/uni/msci/report.pdf, 2008.

[Goel et al, 2008] S. Goel, D. Rao et. al. Plagiarism and its Detection in Programming Languages // http://stanford.edu/~drao/Resources/Plagiarism.pdf , 2008.

[Aho et al., 2007] Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman. Compilers: Principles, Techniques, and Tools. (2nd Edition). Addison-Wesley, CA – 2007г.

Authors' Information

Tetyana Shatovska *– Ass. Ph.D. Prof., Kharkiv National University of radioelectronics; Lenina av. 14, 61166 Kharkiv, Ukraine; e-mail: shatovska@gmail.com*

Major Fields of Scientific Research: Data Mining, Text Classification, Business Intelligence, Artificial Intelligence

Iryna Kamenieva *– Ph. D student., Kharkiv national university of radioelectronics; Lenina av. 14, 61166 Kharkiv, Ukraine; e-mail: iryna.kamenieva@gmail.com*

Major Fields of Scientific Research: Data Mining, Text Classification, Business Intelligence, Artificial Intelligence