Source code similarity

Juraj BRILLA

Slovak University of Technology in Bratislava

Faculty of Informatics and Information Technologies

Ilkovičova 2, 842 16 Bratislava, Slovakia

brilla@fiit.stuba.sk

Nowadays, in the development of any program we can choose from a large number of programming languages. Also, by social networks and the transparency of the Internet our source code can get to potential forgers. Most often various forms of plagiarism occurs in education, where students their solution of assignment share with other schoolmates and they publish on own name without taking proper permission from author or developer. When count of students is large is impossible to check the similarity manually and therefore is necessary to automate this process.

The similarity analyzer must be able to adapt to this trend and the availability of wide variety of programing languages. The analyzer have to be easily extensible for new languages and also performs as accurately analyzes. In addition to ways of analyzing the code is important to look at ways of plagiarism. Edit code can consist of simpler transformations to complicated changes. The simple transformation may be changes in a sequence of stages, changing variable names or supplement comment or insubstantial parts of the source code. This changes are detected relatively easy because a significant part of code remains unchanged. On the other hand the problem with analyzing similarity can occur with code which was changed more complex. It is more complicated to quantify the extent of similarity in the case where functionality of source code has been changed.
Out analyzer use for detection plagiarism method of abstract syntax tree (AST) which are supported by ANTLR tool. This tool can generate parser and lexer for concretely programing language by available grammar. The grammar contain prefixes of program language. In our analyzer are used three grammar for three different program language – Java, CSharp and C language. For extend the additional language platform is necessary to add the grammar of a language. By the lexer and the parser is created AST, which individually pieces of source code defines by abstract concepts. AST is not focused on specific data but scheme of source code. This tree is generated individually for each class, object and interface and by comparing all nodes of tree with other three we get similarity of two compared class, object and interfaces. For speed up counting similarity is this process implemented through a MySQL database. The result similarity of two objects is used for the selection of candidates which are suitable for further comparison. The selection of candidates is applied because of increase efficiency and reduce time consumption of the analyzer. A user can manage the process of selection of candidates through of choice of scale, which can be a maximum of measured value of all source code or an average.

The selected candidates are further analyzed by N-gram method. This method works with source code like with text and the source code is divided into lots of parts with size *n*. These parts are compared with parts of other source codes and the similarity is counted by three algorithms – Jaccard index, Cosine similarity and Levenshtein distance. The results of algorithms are expressed as a percentage and these are output for the user. Each algorithm uses different technic for analyze similarity and for that each result can indicate something else. The user is not required to use all this algorithm but has the option.

Our analyzer of source code similarity is implemented like web application because of easier access and possibility to use this tool without any installation. The web application is realized by TomCat technology and Java servlet and all calculations are performed by Java.