AN EFFICIENT METHOD TO DETECT SOFTWARE PLAGIARISM USING JAVA API

Ms. Chhaya Varade 1, Tejas More 2, Mandar Kulkarni 3, Rohit Nikam 4, Gaurav Patil 5

1Department of Information Technology, GHR Institute of Engineering and Technology, Pune, Maharashtra, India
2Department of Information Technology, G. H. Raisoni Institute of Engineering and Technology, Pune, Maharashtra, India

ABSTRACT:

Open source software allows its usage, modification and redistribution under certain types of licenses. For example, GPL (GNU General Public License) allows users to modify GPL compliance programs freely, as long as the derivative works also follow the tenets of GPL. However, driven by commercial interest some companies and individuals incorporate third-party software without respecting the licensing terms. These intentional or unintentional software license violations lead to serious disputes from time to time.

Software plagiarism refers to computer-assisted plagiarism detection (CaPD) which is an information retrieval (IR) task supported by specialized IR systems, referred to as Plagiarism Detection Systems (PDS).

In this paper we propose a new way to detect software plagiarism using Similarity Measures, which uses adoption of Vector Space Retrieval.

Keywords: Plagiarism Detection Systems, Similarity Measures, Vector Space Retrieval, CaPD.

[1] INTRODUCTION

Plagiarism has become a world-wide problem and is increasing day by day. This problem is getting worse mainly because of the increase in the volume of online publications[8]. Relying only on exact-word or phrase matching for plagiarism detection is not sufficient now. People have started paraphrasing or rearranging words to give a new look to their sentences and thus declare themselves as authors of the material. Using Plagiarism Detection Techniques we can compare a given material with any target material which is either a particular document or in a repository. Different techniques used in the Plagiarism Detection algorithms are discussed in detail here. Here I have given more emphasis on source code related plagiarism[9]. A few case studies show that detection can be done within a large repository. The efficiency and time of the output depends on the algorithms used. Most existing detectors are specially designed to process natural language text or program source code. Systems designed for finding similarities in natural language texts mainly searched the Internet for the possible matches. Text comparisons use simple comparison methods aiming mostly at processing speed and wide coverage. The program source code usually performs a pair wise comparison between single submissions only. Though sophisticated
procedures are being developed which compares with multiple source code programs simultaneously.

[2] LITERATURE SURVEY

The simplest algorithms for plagiarism detection focus around simple general purpose utilities for string searching. For example, using the UNIX utility diff provides a way to compare two files to one another for similarity. Indeed, simply running diff on all pairs of submitted programs would detect L0 plagiarism, however more sophisticated techniques are needed to combat L1 and L2 methods. Heckel's algorithm. [7], provides another tool to measure the similarity of two files. Jankowitz introduced a novel algorithm in [6] for detecting areas of a program to focus on when looking for plagiarism. His method resolves around building a static execution tree for each program then looking for similarities between the trees. If a similarity is found, the two procedures from the source programs reflected in static execution trees are analyzed using the above metric methods for similarity.

[3] PROPOSED WORK

In this technological age, a plagiarism checker is essential for protecting your source code. A plagiarism checker benefits teacher, student, website owner, and anyone else interested in protecting their coding. Plagiarism is a statement that someone copied code deliberately without attribution, and while PDS automatically detects program similarity, it has no way of knowing why codes are similar. It is still up to user to go and look at the part of code that PDS highlights and make decision about whether there is plagiarism or not. There is need to detect such plagiarism, therefore there is need to develop the system for checking the software plagiarism.

Following are the steps we can perform to detect Software Plagiarism:
1. Initially we give .Java class file as input.
2. Using reflect API we extract the basic features of class(Number of Constructor,Parent Class name,Class Is Public or Not.
3. Number of Public Variables, Number of Methods and Respective names etc.)
4. After that we parse the code for Calculating Number of If/For/While etc. Loops.
5. After getting all the Features we compare with stored Feature using Similarity Measures.
Figure: System Architecture

4 ALGORITHM
A. Cosine Similarity

When documents are represented as term vectors, the similarity of two documents corresponds to the correlation between the vectors. This is quantified as the cosine of the angle between vectors, that is, the so-called cosine similarity. Cosine similarity is one of the most popular similarity measure applied to text documents, such as in numerous information retrieval applications [2] and clustering too [4]. Given two documents \( (\text{t}_a) \) and \( (\text{t}_b) \), their cosine similarity is

\[
\text{SIMC}(\text{t}_a, \text{t}_b) = \frac{\text{t}_a \cdot \text{t}_b}{|\text{t}_a| \times |\text{t}_b|},
\]

Where \(\text{t}_a\) and \(\text{t}_b\) are m-dimensional vectors over the term set \(T = \{t_1, \ldots, t_m\}\).
Each dimension represents a term with its weight in the document, which is non-negative. As a result, the cosine similarity is non-negative and bounded between [0, 1]. An important property of the cosine similarity is its independence of document length. For example, combining two identical copies of a document \(d\) to get a new pseudo document \(d_0\), the cosine similarity between \(d\) and \(d'\) is 1, which means that these two documents are regarded to be identical. Meanwhile, given another document \(l\), \(d\) and \(d'\) will have the same similarity value to \(l\), that is, \(\text{sim}(t_d,t_l)=\text{sim}(t_{d'},t_l)\). In other words, documents with the same composition but different totals will be treated identically. Strictly speaking, this does not satisfy the second condition of a metric, because after all the combination of two copies is a different object from the original document. However, in practice, when the term vectors are normalized to a unit length such as 1, and in this case the representation of \(d\) and \(d'\) is the same.

**B. Jaccard Coefficient**

The Jaccard coefficient, which is sometimes referred to as the Tanimoto coefficient, measures similarity as the intersection divided by the union of the objects. For text document, the Jaccard coefficient compares the sum weight of shared terms to the sum weight of terms that are present in either of the two document but are not the shared terms. The formal definition is:

\[
\text{SIM}_J(t_a,t_b) = \frac{|t_a \cap t_b|}{|t_a| + |t_b| - |t_a \cap t_b|}.
\]

The Jaccard coefficient is a similarity measure and ranges between 0 and 1. It is 1 when the \(t_a = t_b\) and 0 when \(t_a\) and \(t_b\) are disjoint, where 1 means the two objects are the same and 0 means they are completely different. The corresponding distance measure is \(D_J = 1 - \text{SIM}_J\) and we will use \(D_J\) instead in subsequent experiments [5].

**[5] EXPECTED RESULT**

In the age of information technologies plagiarism has become more actual and turned into a serious problem. Our proposed system will able to detect plagiarism in software code. In this paper we propose a new way to detect software plagiarism using Similarity Measures, which uses adoption of Vector Space Retrieval. It will extract the features from code and store it in database. Next time when user needs to check plagiarism for code he/she will enter the source code file into software, Software plagiarism detection tool will extract features from given file. In next step it will compare extracted features which are already stored in database. If plagiarism found it will show the content of code that matched with entered code.

**REFERENCES**


[8]. Software Plagiarism Detection with Birthmarks based on Dynamic Key Instruction Sequences Zhenzhou Tian, Qinghua Zheng, Member, IEEE, Ting Liu, Member, IEEE, Ming Fan, Eryue Zhuang and Zijiang Yang, Senior Member, IEEE, 2015


**Author[s] brief Introduction**

1. Ms. Chhaya Varade, Asst Prof., Department of Information technology, GHRIET, Pune
2. Tejas More, Student, Department of Information technology, GHRIET, Pune
3. Mandar Kulkarni, Student, Department of Information technology, GHRIET, Pune
4. Rohit Nikam, Student, Department of Information technology, GHRIET, Pune
5. Gaurav Patil, Student, Department of Information technology, GHRIET, Pune